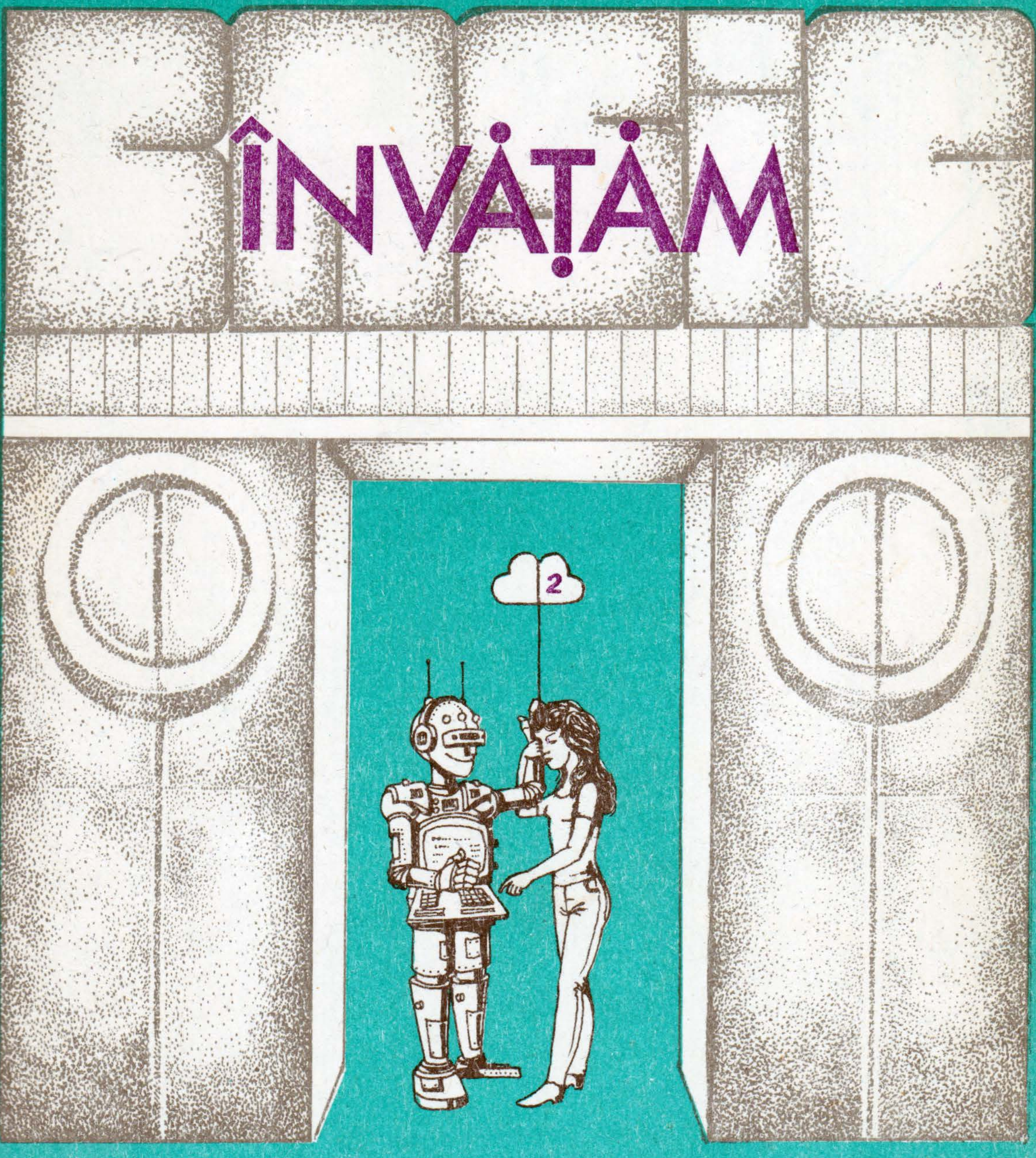
LIVIU DUMITRAȘCU

ÎNVĂȚĂM



MICROELECTRONICA INTERACTIVĂ

EDITURA TEHNICĂ

AUTOMATICĂ
INFORMATICĂ
ELECTRONICĂ
MANAGEMENT

SERIA INIȚIERE

BIBLIOTECA DE ·

Automatică–Informatică–Electronică–Management

SERIA INIȚIERE

LIVIU DUMITRAŞCU a absolvit în 1970 Institutul Politehnic din Bucureşti. În anul 1986 a susţinut teza de doctorat în specialitatea automatizarea instalaţiilor petroliere. Este şef de lucrări la Institutul de Petrol şi Gaze Ploieşti. A activat timp de aproape zece ani la C.E.P.E.C.A. din cadrul Academiei „Ştefan Gheorghiu" unde a predat îndeosebi limbaje de programare în cadrul cursurilor pentru formarea specialiştilor în informatică. A absolvit în Franţa cursuri de specializare în informatică şi tehnică de calcul. A participat la realizarea în colectiv a unor produse-program şi sisteme informatice. A publicat singur sau în colaborare la Editura Academiei R. S. România şi Editura Tehnică mai multe articole şi cărţi, o carte editându-se şi în limba engleză. Cărţile sale din acelaşi ciclu de la Editura Tehnică s-au bucurat de o mare popularitate. Domenii de interes: generarea automată a programelor prin tabele de decizie, calculatoare personale, microbaze de date, inteligenţă artificială.

Prefaţă: ec. NICOLAE BADEA DINCĂ
Recenzii: dr. ing. NICOLAE ŢĂPUŞ
prof. dr. ing. ADRIAN PETRESCU

Redactor: ing. PAUL ZAMFIRESCU

Ideea graficii de copertă şi vignete:
ing. GHEORGHE LUCHIAN
Execuţia coperţii: RADU GHEORGHIAN,
SIMONA DUMITRESCU, BOGDAN ILIE
Desene: arh. CONSTANTIN MIHĂESCU
Tehnoredactor şi machetare: MARIA TRĂSNEA

ISBN 973-31-0010-2
ISBN 973-31-0012-9

Coli de tipar: 23,5. Bun de tipar: 12. VI. 1989.
C.Z. 621.38

Tiparul executat sub com. 260/1988 la
Întreprinderea Poligrafică „Crişana"
Oradea, str. L. Sălăjan nr. 105
Republica Socialistă România

SUMAR

pentru volumele 1 şi 2

VOLUMUL 1

Totul despre BASIC în 14 conversaţii şi . . .

VOLUMUL 2

. . . 7 sinteze

CUPRINS DETALIAT

VOLUMUL 2

. . . 7 sinteze



Paşi printre... conversaţii şi sinteze. Un dialog autor–editor–cititor (III)

□ ... Cu cîteva cunoştinţe noi, pe temelii vechi (Atari, Macintosh, PS/2)

Temeliile sînt date în conversaţii şi sinteze. Aici ne permitem să dăm... o raită prin actualitatea imediată (documentaţii primite de redacţie de la tov. Director NICOLAE BADEA DINCĂ şi de la alţi colaboratori ITCI ş.a.).

Familia ATARI ST*

Mai întîi ne vom referi la trei BASIC-uri, din aceeaşi familie a calculatoarelor personale ATARI ST, în variantele lor recente (1985–1987) şi anume:

1) ST BASIC™–ATARI
2) ATARI ST–MEGA™ COMPUTER
3) OMIKRON BASIC–ATARI ST

variante pe 16 biţi cu microprocesoare din seria 68 000, precedate în trecutul mai îndepărtat de ATARI 400 şi 800, calculatoare personale pe 8 biţi, cu microprocesor 6502 (v. sinteza 21).

1) ST™ BASIC—ATARI (1986—1987)**

este o versiune nouă, de trei ori mai rapidă decît ST BASIC–ATARI, compatibilă cu variantele anterioare, capabilă să rezolve şi tehnicile ferestrelor, meniurilor şi elementelor grafice GEM™.

Cuvinte rezervate

ABS, ALL, AND, AS, ASC, ATN, AUTO
BASE, BLOAD, BREAK, BSAVE
CALL, CDBL, CHAIN, CHRS, CINT, CIRCLE, CLEAR, CLEARW, CLOSE, CLOSEW, COLOR, COMMON, CONT, CONTRL, COS, CSNG, CVD, CVI, CVS
DATA, DEF, DEF FN, DEFDBL, DEFINT, DEFSEG, DEFSNG, DEFSTR, DELETE, DIM, DIR, DO
EDIT, ELIPSE, ELSE, END, EOF, ERA, ERASE, ERL, ERR, ERROR, EQV, EXP
FIELD, FIELD#, FILL, FIX, FLOAT, FOLLOW, FOR, FRE, FULLW
GB, GEMSYS, GET, GET#, GO, GOSUB, GOTO, GOTOXY
HEX$
IF, IMP, INKEY$, INP, INPUT, INPUT#, INPUT$, INSTR, INT, INTIN, INTOUT
KILL

---

* ST BACIC™, Source Book and Tutorial, Atari Corp. Sunnyvale CA 98086, 1986; – Omicron BASIC, Eine Programmiersprache für den ST Computer, Omicron Software, Birkenfeld, 1987; Mega Computer, Atari Corp, S.U.A., 1987.

** 520 ST™ Computer System cu TOS.
(Atari Corp.), GEM (Digital Research),
VT (Digital Equipment Corporation)

LEFT$, LEN, LET, LINE, LINEF, LIST, LLIST, LOAD, LOC, LOF, LOG, LOG 10, LPOS, LPRINT, LSET
MERGE, MID$, MKD$, MKI$, MKS$, MOD
NAME, NEW, NEXT, NOT
OCT$, OLD, ON, OPEN, OPENW, OPTION, OR, OUT
PCIRCLE, PEEK, PELLIPSE, POKE, POS, PRINT, PRINT#, PTSIN, PTSOUT, PUT
QUIT
RANDOMIZE, READ, REM, RENUM, REPLACE, RESET, RESTORE, RESUME, RETURN, RIGHT$, RND, RSET, RUN
SAVE, SEG, SGN, SIN, SOUND, SPACE$, SPC, SQR, STEP, STOP, STR$, STRINGS, SWAP, SYSDBG, SYSTAB, SYSTEM
TAB, TAN, THEN, TO, TRACE, TROFF, TRON
UNBREAK, FOLLOW, UNTRACE, USING
VAL, VARPTR, VDISYS
WAIT, WAVE, WEND, WHILE, WIDTH, WRITE, WRITE#
XOR

Caracterizări

– ST BASIC are trei tipuri de cuvinte rezervate: comenzi (ex. **LOAD**), enunţuri – de program, executabile, neexecutabile – (ex. **CHAIN, PRINT, REM**) – funcţii – de atribuire valori variabilelor.

– Explicarea marii majorităţi a cuvintelor rezervate este posibilă dacă folosim sinteza 15 (îndeosebi cuvintele rezervate ABASIC), dar – mai ales – sintezele 16 (BASIC-AMSTRAD) şi 17 (GW-BASIC de pe Felix PC şi IBM-PC), ca şi sinteza 21 (MBASIC 86).

– Mesajele de erori sînt codificate de la 2–223, mai multe blocuri de coduri (24–29, 34–49, 59–60, 67–98, 112–201, 215–220 fiind pentru erori nedefinite).

– Diferenţe în cadrul limbajelor din familia ST BASIC™ ATARI există; apar – pe măsura perfecţionării sau dotării calculatorului – diferenţe în viteza de calcul, în variabilele de sistem, la menu-uri (sistemul GEM), la ferestre, la lucrul cu „şoricelul", la telecomunicaţii (Terminalul VT 52, modemul RS323).

Menţionăm, în acest sens, calculatoarele MEGA™ Computer Atari, cea mai nouă generaţie din seria ATARI ST, care combină cele mai recente tehnologii de microcalculatoare cu noul sistem de operare TOS™, cu sistemul Graphics Environment Manager (GEM™), care se caracterizează prin:

2) ATARI MEGA (1987): Procesor MC 68 000, 32 bit intern, 16 bit extern, 24 bit magistrală de date, frecvenţa de tact 8 MHz, memorie: **MEGA 2** – 2MB RAM, 192 KB ROM, **MEGA 4** – 2MBRAM, 192 KB. ROM, rezoluţie 640×400 pct. (monocrom), 320×200 pct. (16 culori), 640×200 pct. (4 culori), palete de culori: 512. Interfeţe seriale, paralele, tastatură, modem – RS 232, monitor, audio VT52, dischete 3,5 ţoli/inc, dublă faţă, dublă densitate, disc dur – 10 Mbit/s, cartuş ROM-128 kB, şoricel/joystick sau joystick, imprimantă. Dischetele au capacitatea de 726 016 biţi (ambele feţe) sau 357 376 biţi (o faţă). Cum am spus, sistem de operare TOS™, corelat cu un sistem grafic GEM™ (Graphics Environment Manager), cu meniu, ferestre.

Limbajul de programare este, cum am amintit, o variantă de ATARI BASIC, anume MEGA BASIC, asupra căreia nu mai insistăm, menţionind doar că trebuie să satisfacă şi facilităţile menţionate anterior, cum sînt lucrul cu noile periferice (dischete, şoricel, joystick, teletransmisie, interfaţă audio, disc Winchester) cu mediul grafic GEM, cu tehnica meniurilor.

3) OMIKRON BASIC – este un limbaj al familiei ATARI ST, sub GEM şi TOS, un limbaj rapid, adaptat microprocesoarelor de 16 biţi – 68 000, compatibil în proporţie de 98% cu MBASIC (vezi şi sinteza 21).

Dăm, în continuare, cu puţine precizări, comenzile, instrucţiunile, funcţiile la care trebuie să acordăm o anume atenţie cînd lucrăm în MBASIC sau în OMIKRON BASIC:

c) **LOG** în MBASIC; **LOG**(x) – logaritm natural; **LN**(x) în Omikron BASIC; b) **MKS$** și **MKD$** nu sînt cu totul echivalente; c) **CVS, CVD** la fel; d) **DATA** – diferențe la șiruri de caractere; e) **DEFINT, DEFSNG, DEFDBL, DEFSTR**; **f)** toate variabilele normale sînt în OMIKRON BASIC – long integer, iar variabilele cu virgulă mobilă trebuie prezentate cu postfix-ul !; g) **RANDOMIZE**; h) **RND** (–x); i) **ERASE**; j) **CHAIN... ALL**; k) **PRINT USING**; l) **CLOSE**; m) **INPUT.**

Trebuie să menționăm că ATARI OMIKRON BASIC realizat de autori din R.F.G. (1987, OMIKRON SOFTWARE, Birkenfeld) este adaptat pentru TOS, VTS2 – Standard, Interfață RS 232, sistem GEM DOS, pentru tehnica meniurilor, a sprite-urilor, pentru șoricel, pentru multitasking, cu BIOS (Basic Input/Output System) și XBIOS (X de la extended).

OMIKRON BASIC este un interpretor. Un sistem complet se obține cu OMIKRON BASIC-COMPILER, un compilator pentru calculatoare ATARI ST, care lucrează mult mai rapid (de cca 9,5 ori în medie) decît interpretorul (înmulțirile – de 3,2 ori, buclele repetitive pentru virgulă mobilă de 4,2 ori, buclele repetitive pentru întregi de 27,8 ori, algebra numerelor întregi de 7,4 ori, manipularea șirurilor de 2,1 ori ș.a.).

Compilatorul compilează aproximativ 40 kByte programe BASIC într-un minut, **fiindcă el însuși este un program BASIC de 130 kByte. Programele utilizator: 480 Byte.** Existența acestui compilator caracterizează calculatoarele ATARI ST dotate cu OMIKRON-COMPILER (costul compilatorului ~ 179 DM).

Față de ST™ BASIC-ATARI există o serie de diferențe în plus (la comenzi mai ales), cum sînt **BIOS, XBIOS, CLIP, CVIL, DET** (determinantul unei matrice), **ELLIPSE** (desenează elipse), **FRAC, FORM ALERT, LOT, MOUSEBUT, WVBL.**

*

Numeroase sînt familiile de calculatoare personale pe 8 și 16 biți ca și de calculatoare personal-profesionale pe 16 și 16/32 biți care există pe piața internațională; în ultima perioadă ele s-au dezvoltat în jurul unor microprocesoare din ce în ce mai performante din familiile INTEL, ZILOG, MOTOROLA ș.a. și a unor sisteme de operare ca CP/M (8 biți ca Z80, INTEL 8085), MS/DOS al firmei Microsoft (8/16 bit, ca 8088 sau 16 bit ca 8086), UNIX. Menționăm unele calculatoare de 8/16 sau 16 biți (cu frecvențele de tact), poate insuficient amintite: ACT APRICOT Xi (8086/5), CANON AS (8084/4), DEC RAINBOW 100+(8088/4, Z80A/4), Ericsson PC (8088/4,77), HP 150 (8088/8), IBMPC XT (8088/4,77), IBM PC AT (80 286/6), Olivetti M24 (8086/8), PHILIPS P3100 (8088/4,77), SIEMENS PC-D (80 186/8), SPERRY PC (8088/7,16), TI PC (8088/4,77), WANG PC (8086/8) și APPLE MACINTOSH (68 000/8) la care ne vom referi în continuare.

Familia Macintosh*

Familia Macintosh, bazată pe microprocesoare MOTOROLA 68000, pe 16 biți (sau 16/32 biți) urmașă a familiei de calculatoare personale APPLE (v. și pag. 310, vol. 2) are noi membri: Mac Plus, Mac SE (ce utilizează microprocesoare MOTOROLA 68000 de 8 MHz, cu discuri Winchester și flexibile, cu memorii de la 1 la 2, 4, 6, 8 MB, sistemul SE avînd o viteză mai mare) și Mac II (bazat pe microprocesorul Motorola 68020, de 16 MHz, cu un coprocesor matematic 68881, cu viteze deosebite, rezoluție puternică, adaptat vizualizării în 3D, cu aplicații în CAD/CAM, proiectare grafică, arhitectură) aflați în luptă de piață cu familia IBM PC și cu calculatoarele firmei Compaq. Aceasta se traduce și prin compatibilizări realizate la nivelul sistemelor de operare, al interfețelor ș.a. Astfel că, începînd din 1987 familia Macintosh are 4 membri (**Macintosh 512E,** cu 512 KRAM, disc flexibil intern de 800 K, monitor monocrom, posibilitate de dotare cu șoricel și imprimantă, două porți seriale, un conector de disc flexibil, extern etc. – 1699 $; **Macintosh Plus,** memorie expandabilă de la 1 MB (2 199 $). **Macintosh SE,** memorie expandabilă de la 1 la 4 MB, opțiuni de două discuri flexibile interne de 800 K sau un disc flexibil de 800 K și un disc Winchester de 20 MB ambele interne (2 769–3 569 $) și **Macintosh II** cu monitor color, cu memorie expandabilă de la 1 la 8 MB, cu memorii interne Winchester de

[1] Trend profil EXTRA, 1/1985, Wien.
[2] Seria AMC, Editura Tehnică, 1985–1989.
* Mac User, vol. 3, nr. 9, sept. 1987, Apple Computer, S.U.A.

40–80 MB etc. (3 769–5 369 $) în afara celor portabile (Dina-Mac, cu ecran electroluminiscent și Lap-Mac cu display cu plasmă. Toate pot lucra în rețele cu IBM PC-uri, cu alte sisteme ce funcționează în MS-DOS standard. Limbajele BASIC utilizate sînt compatibile unilateral cu Microsoft BASIC, care a fost primul limbaj de programare al familiei MAC, ca interpretor. Acum familia Macintosh utilizează și compilatoare BASIC Microsoft. De curînd, se folosesc și produsele rapide ale firmei Borland (ca Turbo-BASIC etc.).

Familia IBM PS/2*

Cel mai recent sistem de calculatoare personal-profesionale al IBM, realizat pe 16 biți (modelul 30 — cu microprocesorul 8086, modelele 50–60, cu microprocesorul 80286) și **modelul 80, cu microprocesorul 80386 de 32 biți → 25 MHz)** este sistemul IBM PS/2, cu sistemele de operare PC DOS 3.3 și OS/2. Modelul 30 este compatibil la nivel BIOS cu IBM PC-XT (un recent produs românesc compatibil cu IBM PC-XT este calculatorul Junior al firmei Ieper). Sistemul de operare PC-DOS 3.3 este o perfecționare a sistemului DOS 3.2.

Sistemul de operare OS/2 (o nouă generație de DOS, cu o mare dezvoltare în 1989), este un sistem multitasking, ce adresează pînă la 16 MO de memorie, cu puternice facilități, de ex. cu versiuni pentru pachete de manipulare a bazelor de date (Paradox OS/2 a firmei Borland, R: BASE pentru OS/2 a firmei Microsim, Q&A OS/2 a Corporației Symantec) comparabile cu DBMS-uri (SGBD-uri) ca Informix-SQL, XQL, Oracle, Ingres, dBase IV. Compatibilități cu sistemele UNIX performante (de exemplu cu UNIX System V) (Modelul 30 costă cam 1 950 $, 50–60 cam 2 500–3 300 $, modelul 80 cam 4 300–6 000 $, la care se adaugă perifericele — imprimante, modemuri etc.). Menționăm aici și prețurile practicate pentru limbaje: MS Basic compiler ~ 176 $, Quick Basic 4.0 ~ 59 $, Turbo BASIC ~ 62 $ (vezi mai jos).

Sistemul de calcul personal-profesional IBM PS/2 modelul 80, este îndreptățit să fie liderul microcalculatoarelor personal-profesionale de 32 biți (Trei noi standarde video MCGA, VGA . . . ; noi monitoare analogice, compatibilitate cu noul BIOS, nouă magistrală "Micro-Channel", OS/2 perfecționat).

Turbo-Basic pentru IBM PC și compatibile.** Turbo-Basic este un mediu de programare pentru calculatoare personale (IBM-PC) și compatibile. Este o combinație de editor, compilator rapid (totul în memoria internă), bibliotecă "run-time" și link-editor intern. Interfața utilizator este modernă (cu ferestre și meniuri tip "stor" (pull-down)). Limbajul este puternic structurat (blocuri IF/THEN/ELSEIF/ELSE/ENDIF, DO/LOOP, CASE/SELECT, CALL/SUB). Numerotarea liniilor este înlocuită cu etichete.

Se leagă ușor cu limbajul de asamblare și oferă numeroase directive compilator pentru: compilare condiționată, tratare erori, controlul bufferelor și altele.

Editorul este de tip Word Star, comod pentru toți cei obișnuiți cu acest produs, dar și cu alte produse ale firmei BORLAND (Turbo Pascal, Turbo C, Turbo ASM, Turbo Editor etc.).

Alte caracteristici distinctive sînt: suportă "virgula mobilă" (fie pentru coprocesor 8087, fie prin emulare soft); suportă videoterminale EGA ("Enhaced Graphical Adapter); tipul de dată "string" suportă lungimi pînă la 32 767 caractere.

---

* PC Magazine, vol. 8, number 11, iune 13, 1989, Ziff Davis Publishing Company, New York, S.U.A.

— A. Petrescu ș.a. ABC de calcul electronic . . . și nu doar atît; Editura Tehnică, 1989 (sub tipar).

** Comunicare personală, Gh. Răuț, ITCI, 1989.

□ Şi cu . . . VAX-uri (mini, supermini?!)

Familia VAX*

Familia VAX contează drept "cel mai mare număr de vînzări" dintre minicalculatoarele de 32 biţi. Ea continuă familia PDP, introdusă de DEC în 1970, cu produse de la MicroVAX, care costă ~ 20 000 \$, pînă la VAX8650, **supermini**, care – în configuraţie completă – costă în jur de 1 milion \$. Produsele familiei sînt compatibile software; sistemul de operare VAX/VMS. (Virtual Memory Operating System) rulează pe toate procesoarele VAX, care suportă şi versiunea ULTRIX-32 a sistemului de operare UNIX. Arhitectura sistemului VAX (modelele Micro VAX I, II, VAX 11/725, 730, 750, VAX 11/780, 782, 785, VAX 8600, 8650) este diferită de a lui PDP-11, dar permite execuţia tuturor programelor scrise pentru PDP-11 şi compatibile. Capacitatea de memorie de la 3 MB la 68 MB. Memoria virtuală de 4 gigabytes. Modelele mai mici au un UNIBUS, cele mai mari au un UNIBUS adiţional şi mai multe MASSBUS-uri. Modelele perfecţionate au viteze mai mari de procesare. Numărul de utilizatori creşte de la 4 – pentru Micro VAX I, la 8 pentru VAX 11–725, la 24 pentru VAX 11–730, la 100 interactivi pentru VAX 11–780 şi la cîteva sute pentru VAX 8600 şi 8650. Limbajul de asamblare este VAX MACRO.

Menţionăm o comparaţie* între timpii de procesare (în µs) pentru operaţii tipice, în cazul unui microprocesor (Intel 80286), al minicalculatorului PDP 11/44 şi al minicalculatorului VAX 11/780 (din care . . . rezultă multe).

Operaţie	Intel 80286	PDP 11/44	VAX 11/780	Erori detectate
Adunări întregi, scăderi	0,875	1,4	0,6	depăşire aritm.
Multiplicări întregi	3,0	6,6	1,0	depăşire aritm.
Comparări întregi	0,875	1,4	0,6	depăşire aritm.
SAU logic	0,875	0,36	0,2	nu
Adunări în virgulă mobilă	nu	8,9	2,0	depăşire

Revenim rapid la BASIC-uri pe minicalculatoarele CORAL, INDEPENDENT (compatibile cu PDP 11) ca să ajungem în fine la VAX BASIC*.

BASIC PLUS (BPL) este un limbaj de nivel înalt pentru aplicaţii interactive în domenii ştiinţifice, economice, în învăţămînt. Interpretorul asociat, ce extinde masiv specificaţiile limbajului BASIC Darmouth include opţiuni de compilare pentru obţinerea unor module obiect BASIC-PLUS şi permite utilizarea procesorului de virgulă mobilă prezent pe minicalculatoarele I-102 F, I-106, CORAL 4021.

BASIC PLUS 2 (BP2) extensie a lui Basic-Plus pentru aplicaţii sofisticate de culegere şi prelucrare date în domenii ştiinţifice şi comerciale. Compilatorul asociat permite utilizarea unor construcţii structurate la nivel de bloc şi a unor metode de acces RMS pentru adresare secvenţială, relativă şi indexată a fişierelor.

În conjuncţie cu programele de aplicaţie scrise în limbajele BASIC-PLUS 2 se utilizează FMS (Subsistem conversaţional pentru interfaţarea programelor de aplicaţie cu facilităţi interactive, orientate spre ecrane). Accesul la reţeaua MININET/MIX este permis programelor utilizator scrise şi în acest BASIC PLUS 2.

* – Schneider, Davis, Mertz, Computer Organization and Assembly Language Programming for the VAX, John Wiley, 1987.

– A. Davidoviciu ş.a. **Sistemul de operare MIX şi limbajul său MACRO,** Editura Tehnică, 1989 (în manuscris).

Elaborarea unui sistem de operare MIX/VMS pentru minicalculatoare româneşti de 32 biţi compatibile VAX 11 este obiect de perspectivă 1987–90 al ITCI.

VAX BASIC beneficiază de un mediu de programare cu înaltă interactivitate, combină facilităţile unui BASIC structurat, compilat, cu ale limbajului RSTS/E BASIC-PLUS, cu performanţele atinse de un limbaj VAX care este complet integrat în **mediul de dezvoltare VMS.** Limbajul VAX BASIC este un limbaj cu o înaltă extindere a implementărilor. Dispune de o matematică puternică şi de facilităţi de manipulare a şirurilor, de suport pentru numele variabilelor simbolice/depanare, are un RMS complet indexat secvenţial şi operaţii I/0 relative. VAX BASIC poate fi utilizat atît ca **interpretor** cît şi drept **compilator.**

☐ Mai facem cîteva . . . bucle printre conversaţii şi . . . STOP cadru

Încă puţin despre . . . tehnica meniurilor

În conversaţiile 8, 9, 10, 11, 12, 13 şi 14 a fost prezentată şi aplicată tehnica meniurilor, mult folosită în dialogul (activ) al utilizatorului cu sistemele de calcul. Dacă cei mai puţin experimentaţi o consideră de neînlocuit, cei experimentaţi opinează pentru nefolosirea ei, considerînd că . . . deranjează. În consecinţă, cum trebuie procedat? În vederea organizării dialogului cu sistemele de calcul prin ecrane pur informaţionale se recomandă:

- să se furnizeze secvenţe de interacţiune simple, consistente;
- să se evite încărcarea minţii utilizatorului cu prea multe opţiuni şi stiluri de comunicare cu calculatorul;
- utilizatorul **mai puţin experimentat,** să fie îndrumat la fiecare etapă a interacţiunii, corelat cu posibilitatea eliminării dinamice a secvenţelor de îndrumare pentru **utilizatorii experimentaţi;**
- utilizatorul să primească o reacţie (grafică/alfanumerică) la fiecare comunicare cu sistemul.

Pentru utilizatorii meniurilor aflate pe tabletă sau digitizor (v. **vol. 2, sinteza 18**) se sugerează următoarele:

- se va evita utilizarea în cadrul unui meniu a unui număr mare de cîmpuri de meniu;
- se va urmări gruparea unor funcţii înrudite logic, evitîndu-se deplasarea miinii ce manevrează instrumentul de digitizare pe distanţe prea mari;
- nu se vor concepe meniuri cu cîmpuri de dimensiuni mici (sînt greu de aliniat);
- se vor folosi diverse culori sau stiluri de scriere pentru a evidenţia zonele de interes pentru utilizator;
- dacă uneori scrisul ar ocupa un text prea lung se va utiliza un simbol cît mai reprezentativ;
- se va evita ca asamblarea unei comenzi să se facă pentru un număr prea mare de secvenţe de culegeri.

De la grafica pasivă la grafica interactivă

Începînd cu anul 1970 **funcţiile de bază ale echipamentelor şi sistemelor grafice depindeau atît de clasa de aplicaţii abordată cît şi de calculatorul gazdă.** S-a ajuns ca, la un moment dat să existe mai multe şcoli de grafică ce-şi disputau abordările. Aşa au apărut, cu avantaje şi dezavantaje, limbajele de programare pentru grafică, extensiile limbajelor de programare de nivel înalt cît şi pachetele de rutine grafice apelabile dintr-un limbaj de nivel înalt.

Din anul 1975 experții în domeniul graficii au inițiat o activitate de standardizare în grafică urmărind obiective ca: portabilitatea programelor de aplicație, independența programelor de aplicație față de echipamentele grafice utilizate, portabilitatea informațiilor grafice, portabilitatea instruirii etc.

Apare deci ca necesară o aliniere a tuturor instrumentelor de grafică la standardele de grafică curente: GKS, PHIGS, CGI, CGM, IGES etc. (v. de exemplu, realizările CEPECA – GPL, FEX GKS aliniate la GKS și realizările ITCI). Cît privește proiectarea asistată de calculator, ea există practic de cînd s-a realizat primul calculator, dar se pare că acum se pune problema în mod mai sistematic.

Jocuri cu calculatorul

Se știe bine că jocul ajută la instruire. Jocurile copiilor, după cum remarca M. Montegne **"nu sînt deloc jocuri și este mai corect ca ele să fie considerate cea mai importantă și profundă preocupare a acestei vîrste".** Calculatorul personal a fost primul instrument individual care a permis ca oameni de toate vîrstele să-l folosească nu numai pentru lucruri . . . serioase ci și pentru jocuri.

Jocurile – "drojdie distractivă" a calculatoarelor personale – sînt abordate in lucrarea noastră (vol. 2, sinteza 19) din punct de vedere al proiectării și implementării acestora. Vă prezentăm totodată și 25 programe sursă de jocuri în BASIC, pentru calculatoarele HC-85, TIM S, SPECTRUM, AMSTRAD și COMMODORE.

În sprijinul dezvoltării conceptelor de analiză și proiectarea structurată

În ultimul deceniu conceptul de programare a evoluat considerabil, în sensul fundamentării teoretice și al sistematizării sale. Această dezvoltare are legături intime cu progresul înregistrat în aplicarea concepțiilor sistemice în întreg domeniul sistemelor/produselor informatice. Numeroase tehnici au fost avansate în vederea ridicării nivelului calitativ al sarcinilor prestate, a reducerii duratelor de proiectare și construire a sistemelor/produselor informatice, a asigurării unei fiabilități și economicități sporite. Astfel se explică apariția în ultimul timp pe plan mondial a peste 40 de metodologii structurate. În general, **metodologiile structurate sînt orientate către structuri de date, fluxuri de date** etc. și folosesc descrieri preponderent grafice.

În ceea ce ne privește am utilizat **metodologia Jackson** orientată spre structuri de date, pe care am aplicat-o pe cazuri simple, ce ne-au servit ca pretext pentru introducerea elementelor de limbaj FORTRAN/COBOL/BASIC în toate cele trei lucrări din ciclul "Învățăm . . . (limbajul) conversînd cu calculatorul".

Recomandăm cititorului și alte metodologii de analiză și proiectare structurată, ca de exemplu cea aparținînd lui Yourdon și Constantine, metodologie ce am constatat că este foarte bine însușită și răspîndită de către specialiștii CEPECA. În acest sens se pot consulta lucrările mai importante*.

* * *

După salturile pe care le-ați făcut urmărind cele 3 grupuri (I, II, III) de pași (nu prezintă interes prea mare în ce moment al studiului ați . . . sărit, totul este să o faceți), vă recomandăm să luați conversațiile și sintezele de la început. Asta pentru că, parafrazîndu-l întrucîtva pe N. Armstrong . . . cînd a pășit pe Lună ("un mic pas pentru mine, un pas uriaș pentru omenire"), micii pași ai fiecăruia din dumneavoastră pentru a conversa cu aceste mașini ce par ale altei planete, vă permit a . . . încerca să țineți pasul, cu pașii uriași ai omenirii, făcuți – după cum știți – și în această direcție.

Iar acum dăm din nou . . . cuvîntul cititorului, nu fără a-i ura, încă odată . . . mult succes.

ÎI MAI RECOMANDĂM, ÎN FINE, SĂ CONSULTE ȘI ALTE CĂRȚI (v. pag. XV–XVI).

* Yourdon, E., & Constantine, L. L., **Structured Design,** Prentice-Hall, 1979 și Colectiv 100 autori, **Ingineria de sistem, automatică și informatică în transporturi,** vol. 2, cap. 11, Editura Tehnică, 1989.

CĂRŢI APĂRUTE ÎN 1986–89
CARE SE MAI POT AFLA ÎN REŢEAUA DE LIBRĂRII

Biblioteca de automatică, informatică, electronică, management

Nr.	Autori	Titlu
1.	Adrian Davidoviciu, Boldur Bărbat	**Limbaje de programare pentru sisteme în timp real**, 224 pag., 23 lei.
2.	Marius Guran, Florin Filip	**Sisteme ierarhizate în timp real, cu prelucrare distribuită a datelor**, 296 pag., 29 lei.
3—4.	Cr. Giumale, D. Preoţescu, L. D. Serbănaţi, D. Tufiş, Gh. Tecuci, D. Cristea	**LISP. Programare (DM—LISP). Probleme rezolvate. Aplicaţii complexe LISP 86 pentru micro (Felix PC, M 216) şi TC LISP pentru minicalculatoare (CORAL/Independent). Sisteme de I. A. pe TC Lisp** 2 vol., 44 lei.
5.	M. Suciu, D. Popescu Tr. Ionescu	**Microprocesoare, microcalculatoare şi roboţi în automatizări industriale**, 384 pagini, 28 lei.
6—7.	T. Baron, Al. Isaic-Maniu, L. Tövissi, D. Niculescu, C. Baron, V. Antonescu, I. Roman	**Calitate şi fiabilitate, manual practic**, 2 vol., 1100 pag. **(cu 300 aplicaţii, exemple, studii de caz, 4 rigle de calcul, 5 standarde, 2 000 teste-întrebări rezolvate)**, 139 lei.
8.	Gh. Turbuţ, I. Boicu, E. Spirea, M. Huţanu, I. Tomescu şi colectiv 100 specialişti din MTTc, ITCI, IPA	**Inginerie de sistem, automatizări şi informatică în transporturi feroviare, navale, aeriene, rutiere**, vol. 1, 758 pag., 75 lei.
9—12.	Colective largi	**Automatică, management, calculatoare** (AMC) volumele 52—55 ~2000 pag. ~180 lei (a se căuta şi volumele anterioare în curs de epuizare).
13.	T. Geber, V. Cristea, V. Săvescu, I. Miu, R. Bulgakov, M. Vuici	**Echipamente periferice**, vol. 3, 260 pag., 19 lei.
14—15.	Gh. Sabău, Al. Sotir şi colectiv 11 specialişti ASE, CSP, ITCI, Centrul de Calcul teritorial Constanţa	**Practica bazelor de date. Totul despre... SOCRATE şi SOCRATE — MINI pe Felix C, CORAL, INDEPENDENT.** Volumele 1 şi 2, 768 pagini, Seria Practică. Lei 62.
16.	Gh. Turbuţ, E. Spirea, M. Huţanu, I. Tomescu, I. Boicu şi colectiv 100 specialişti MTTc, ITCI, IPA, IPB, Centrul de Calcul Teritorial Constanţa	**Inginerie de sistem, automatizări şi informatică în transporturi feroviare, navale, aeriene, rutiere.** Volumul 2, 1 000 pagini, Seria Fundamente, 85 lei.
17.	N. Patrubani	**Totul despre... microprocesorul Z80**, vol. 1 şi vol. 2, 700 pagini, (o parte a tirajului însoţită de o casetă pentru un simulator al funcţionării microprocesorului pe calculatoarele personale PRAE şi aMIC), 140 lei cu casetă, fără casetă 65 lei.

CĂRȚI ÎN CURS DE APARIȚIE

Biblioteca de automatică, informatică, electronică, management

1—2. L. Dumitrașcu	**Invățăm microelectronică interactivă. Totul despre ... BASIC în 14 conservații și 7 sinteze pe Felix C, CORAL, INDEPENDENT, Felix PC, M118, TPD, HC. 85, aMIC, COMMODORE, AMSTRAD și compatibile,** volumele 1 și 2, 980 pag., lei 100, apare în trim. III.
3— 4. A Petrescu și colectiv IPB, ITCI, Fabrica de calculatoare, Liceul Dimitrie Cantemir, CNOP	**ABC de calcul electronic. Totul despre ... HC85,** vol. 1 și vol. 2, 700 pagini **(o parte a tirajului cu 2 casete cu programe, acționînd calculatoare personale HC85 și compatibile SINCLAIR SPECTRUM),** 250 lei cu casete, 70 lei fără casete.
5—6. A. Tănăsescu și colectiv IPB, ITCI, ISPIF	**Grafică asistată de calculator. Programe Fortran pe minicalculatoare, pentru reprezentări geometrice,** vol. 1 și vol. 2, 800 pag., 80 lei, apare în trim. III.
7—13. Colective largi	**Automatică, management, calculatoare (AMC). Serie continuă de instruire, informare, sinteze, cercetări aplicative în sisteme electronice, automate, informatice, de conducere.** Volumele 56—61. Volume de ~ 350 pag. și ~ 35 lei fiecare. **Echipamente electronice și tehnică de calcul-manuale de utilizare. Calculatoare personale, programe. Aplicații informatice în ramuri industriale. Proiectarea asistată de calculator. Grafică interactivă. Automatizarea și informatizarea proceselor. Conducere și organizare asistată. Limbaje și produse program,** ș.a.m.d.
14—15. A. Davidoviciu și colectiv ITCI, ASE	**Sistemul de operare MIX și limbajul MACRO pentru minicalculatoarele CORAL/INDEPENDENT,** 2 volume, 800 pagini, 90 lei.
16—17. I. Văduva, V. Baltac, Florescu V. și colectiv ASE, ITCI	**Informatică economică,** 2 volume, 800 pag. Preț 80 lei
18. Rusu O., Brudaru I.	**Proiectarea liniilor flexibile. Echilibrarea (concepte, modele, algoritmi de tip ALGOL, programe COBOL și Fortran).** 300 pag., 30 lei.
19. P. Constantinescu	**Sinergia, informația și geneza sistemelor,** 350 pag., 35 lei.

Se difuzează prin unitățile centrelor de librării, spre care se îndrumă întreprinderile și cititorii.

PENTRU ACESTE CĂRȚI SE POT FACE, TOTUȘI, ȘI COMENZI FERME LA EDITURA TEHNICĂ, PIAȚA SCÎNTEII 1, BUCUREȘTI.

Comenzile întreprinderilor se semnează de director și contabil șef, cele ale cititorilor individuali au indicată adresa exactă. Comenzile se trimit de editură la centrele de librării, cu indicarea unor priorități de satisfacere a lor. Plata nu se face decît la primirea exemplarelor de la rețeaua de librării.

...7 SINTEZE



Cuvinte rezervate și diagnostice ...post mortem BASIC

☐ BASIC-aMIC

Cuvinte rezervate

ABS, AT, ATN
CALL, CHR$, CON, COS
DATA, DIM, DRAW
END, EXP
FOR
GET, GOSUB, GOTO, GSINPUT
IDN, IF, INPUT, INT, INV, INIT, INKEY$
LET, LEN, LIST, LOG, LOAD
MAT, MOVE
NEXT
ON
PLOT, PRINT, PUT
READ, REM, RESTORE, RETURN, RND,
RUN, RMOVE, RDRAW, ROTATE
SAVE, SGN, SIN, SQR, STEP, STR$, STOP,
SCR, SCALE
TO, TAN, THEN, TRN
UNPLOT
VAL, VIEWPORT
ZER
WINDOW

Mesaje de erori

Cod	Mesaj
01	programul nu se termină cu instrucțiunea **END** sau **STOP**
02	tip de instrucțiune nerecunoscut
03	există instrucțiuni sursă după instrucțiunea **END**
04	numărul liniei destinație incorect (în instrucțiunile **IF, GOTO, GOSUB, ON**)
05	numărul liniei destinație inexistent
06	caracter ilegal
07	instrucțiune neterminată
08	expresie incorectă
09	eroare la conversia în virgulă mobilă
10	utilizare incorectă a funcțiilor **GET** sau **PUT**
11	tentativă de redimensionare (cu **DIM**) a unui tablou
12	tablou utilizat înainte de a fi definit
13	argumentul funcțiilor **SIN, COS, TAN** prea mare ($>10^6$)
14	relație incorectă în instrucțiunea **IF**
15	expresie prea complicată (depășirea stivelor)
16	eroare în ridicarea la putere (0 la puterea 0 sau număr negativ la putere reală)
17	instrucțiune **FOR** fără instrucțiunea **NEXT** corespunzătoare
18	instrucțiune **NEXT** fără instrucțiunea **FOR** corespunzătoare

19 depăşire stivă **FOR** (peste 3 cicluri cuprinse unul într-altul)
20 indice mai mare de 254 sau egal cu ∅
21 se doreşte citirea mai multor constante decît au fost definite prin instrucţiunea **DATA**
22 depăşirea valorilor declarate ale indicilor
23 radical din număr negativ
24 logaritm din număr negativ
25 dimensiuni incompatibile pentru produs de matrici
26 matricea din membrul stîng al egalităţii nu poate apărea în membrul drept (pentru transpusă sau produs de matrici)
27 matrice singulară (nu poate fi inversată)
28 redimensionare incorectă (noile dimensiuni depăşesc pe cele maxime)
29 prin redimensionare nu poate fi schimbat numărul de dimensiuni ale unui tablou
30 matricea nu este pătrată
31 matricile nu au aceleaşi dimensiuni (pentru sumă sau diferenţă)
32 subrutina, în limbaj de asamblare, cu numărul dorit nu a fost găsită în tabela de subrutine
33 parametrii incorecţi în instrucţiunea de prelucrare grafică (**VIEWPORT, WINDOW, SCALE**).

□ BASIC-PRAE

Cuvinte rezervate

ABS, ALOAD, AMERGE, AND, ASAVE, ASC, AT, ATN, AUTO

BEEP

CALL, CLEAR, CLS, CHR$, CIRCLE, CONT, COS

DATA, DEFFN, DELETE, DIM, DRAW, DRAWC

EDIT, ELSE, END, EXP

FN, FNEND, FOR, FRE

GOSUB, GOTO

IF, INP, INPUT, INSTR, INT

KILL

LET, LEFT$, LEN, LINE, LISTEN, LIST, LLIST, LNULL, LOG, LPRINT, LOAD, LVAR, LTRACE, LWIDTH

MID$

NEXT, NEW, NOT, NULL

ON, OR, OUT

PEEK, PLOT, PLOTC, POKE, POS, PRINT, PRECISION

RANDOMIZE, READ, REM, RENUMBER, RESTORE, RETURN, RIGHT$, RND, RUN

SAVE, SGN, SIN, SQR, SPC, STEP, STR$, STOP, SWITCH

TAB, TO, TAN, THEN, TRACE

USING

VAL

WIDTH

Mesaje de erori

NEXT W/O FOR
SYNTAX ERROR (IN LINE n)
RETURN W/O GOSUB
OUT OF DATA
ILLEGAL FUNCTION
ARITHMETIC OVERFLOW
OUT OF MEMORY
UNDEFINED STATEMENT
SUBSCRIPT OUT OF RANGE
RE-DIMENSIONED ARRAY
ILLEGAL DIRECT
TYPE MISS-MATCH
NO STRING SPACE
STRING TOO LONG
TOO COMPLEX
CAN'T CONTINUE
TAPE ERROR
FNRETURN W/O FUNCTION CALL
MISSING STATEMENT NUMBER
* INVALID INPUT
* EXTRA LOST

☐ BASIC HC-85, TIM S, SPECTRUM

Cuvinte rezervate

ABS, ACS, AND, ASN, AT, ATN, ATTR

BEEP, BIN, BORDER, BREAK, BRIGHT

CHR$, CIRCLE, CLEAR, CLOSE#, CLS, CODE, CONTINUE, COPY, COS

DATA, DEFFN, DELETE, DIM, DRAW

EDIT, ENTER, EXP

FLASH, FN, FOR

GOSUB, GOTO

IF, IN, INK, INKEY$, INPUT, INT, INVERSE

LEN, LET, LINE, LIST, LLIST, LLIST#, LN, LOAD, LPRINT, LPRINT#

MERGE

NEW, NEXT, NOT

OPEN#, OR, OUT, OVER

PAPER, PAUSE, PEEK, PI, PLOT, POINT, POKE, PRINT, PRINT#

RANDOMIZE, READ, REM, RESTORE, RETURN, RND, RUN

SAVE, SCREEN$, SGN, SIN, SQR, STEP, STOP, STR$

TAB, TAN, THEN, TO

USR

VAL, VAL$, VERIFY

Mesaje de erori

0	OK
1	NEXT without FOR
2	Variable not found
3	Subscript wrong
4	Out of memory
5	Out of screen
6	Number too big
7	Return without GOSUB
8	End of file
9	Stop statement
A	Invalid argument
B	Integer out of range
C	Nonsense in BASIC
D	BREAK–CONT repeats
E	Out of DATA
F	Invalid file name
G	No room for line
H	STOP in INPUT
I	FOR without NEXT
J	Invalid I/O device
K	Invalid colour
L	BREAK into program
N	RAMTOP no good
M	Statement lost
O	Invalid stream
P	FN without DEF
Q	Parameter error
R	Tape loading error

☐ BASIC-AMSTRAD

Cuvinte rezervate

ABS, AFTER, AND, ASC, ATN, AUTO

BIN$, BORDER, BREAK

CALL, CAT, CHAIN, CHR$, CINT, CLEAR, CLG, CLOSEIN, CLOSEOUT, CLS, CONT, COPYCHR$, COS, CREAL, CURSOR

DATA, DEC$, DEF, DEFINT, DEFREAL, DEFSTR, DEG, DELETE, DERR, DI, DIM, DRAW, DRAWR

EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE, ERL, ERR, ERROR, EVERY, EXP

FILL, FIX, FN, FOR, FRAME, FRE

GOSUB, GOTO, GRAPHICS

HEX$, HIMEM

IF, INK, INKEY, INKEY$, INP, INPUT, INSTR, INT

JOY

KEY

LEFT$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG 10, LOWER$

MASK, MAX, MEMORY, MERGE, MID$, MIN, MOD, MODE, MOVE, MOVER

NEXT, NEW, NOT

ON, ON BREAK, ON ERROR GOTO 0, ON SQ, OPENIN, OPENOUT, OR, ORIGIN, OUT

PAPER, PEEK, PEN, PI, PLOT, PLOTR, POKE, POS, PRINT

RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM, RESTORE, RESUME, RETURN, RIGHT$, RND, ROUND, RUN

SAVE, SGN, SIN, SOUND, SPACE$, SPC, SPEED, SQ, SQR, STEP, STOP, STR$, STRING$, SWAP, SYMBOL

TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, TIME, TO, TROFF, TRON

UNT, UPPER$, USING

VAL, VPOS

WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE

XOR, XPOS

YPOS

ZONE

Mesaje de erori

1 Unexpected NEXT
2 Syntax Error
3 Unexpected RETURN
4 DATA exhausted
5 Improper argument
6 Overflow
7 Memory full
8 Line does not exist
9 Subscript out of range
10 Array already dimensioned
11 Division by zero
12 Invalid direct command
13 Type mismatch
14 String space full
15 String too long
16 String expression too complex
17 Cannot CONTinue
18 Unknown user function
19 RESUME missing
20 Unexpected RESUME
21 Direct command found
22 Operand missing
23 Line too long
24 EOF met
25 File type error

□ BASIC-COMMODORE

Cuvinte rezervate

ABS, AND, ASC, ATN

CHR$, CLR, CLOSE, CMD, CONT, COS

DATA, DEF, DIM

END, EXP

FN, FOR, FRE

GET, GET#, GOSUB, GOTO

IF, INPUT, INPUT#, INT

LEFT$, LEFT, LEN, LET, LIST, LOAD, LOG

MID$

NEW, NEXT, NOT

ON, OPEN

PEEK, POKE, POS, PRINT, PRINT#

READ, REM, RESTORE, RETURN, RIGHT$, RND, RUN

SAVE, SGN, SIN, SPC, SQR, STEP, STOP, STR$, SYS

TAB, TAN, THEN, TI$, TI

USR

VAL, VERIFY

WAIT

Mesaje de erori

BAD DATA
BAD SUBSCRIPT
CAN'T CONTINUE
DEVICE NOT PRESENT

DIVISION BY ZERO
EXTRA IGNORED
FILE NOT FOUND
FILE NOT OPEN
FILE OPEN
FORMULA TOO COMPLEX
ILLEGAL DIRECT
ILLEGAL QUANTITY
LOAD
NEXT WITHOUT FOR
NOT INPUT FILE
NOT OUTPUT FILE
OUT OF DATA
OUT OF MEMORY
OVERFLOW
REDIM'D ARRAY
REDO FROM START
RETURN WITH-OUT GOSUB
STRING TOO LONG
SYNTAX
TYPE MISMATCH
UNDEF'D FUNCTION
UNDEF'N STATEMENT
VERIFY

□ BASIC-80

Cuvinte rezervate

ABS, ASC, ATN, AUTO, AND

BASE

CALL, CDBL, CHAIN, CHR$, CINT, CLEAR, CLOSE, COMMON, CONT, COS, CSNG, CVI, CVS, CVD

DATA, DEF, DEFINT, DEFSNG, DEFDBL, DEFSTR, DEF USR, DELETE, DIM, DRAW

EDIT, ELSE, END, EOF, ERASE, ERR, ERL, ERROR, EXP, EQV

FIELD, FIX, FOR, FRE, FN

GET, GOSUB, GOTO

IF, INKEY, INSTR, INP, INPUT, INPUT#, INT, IMP

KILL

LEFT$, LET, LEN, LINE, LIST, LLIST, LOAD, LOC, LOG, LPRINT, LPOS, LSET

MERGE, MID$, MKI$, MKS$, MKD$, MOVE , MOD

NAME, NEXT, NEW, NULL, NOT

OCT$, ON, OPEN, OPTION, OUT, OR

PEEK, POKE, POS, PRINT, PRINT#, PUT

RANDOMIZE, RDRAW, READ, REM, RENUM, RESTORE, RESUME, RETURN, RIGHT$, RMOVE, RND, ROTATE, RSET, RUN

SAVE, SGN, SIN, SPACE$, SPC, SQR, STOP, SWAP, STR$, STRING$

TAB, TAN, THEN, TRON, TROFF

USING, USR

VAL, VARPTR, VIEWPORT

WAIT, WHILE, WEND, WIDTH, WINDOW, WRITE, WRITE# XOR

Mesaje de erori

NEXT without FOR
Syntax error
RETURN without GOSUB
Out of DATA
Illegal function call
Overflow
Out of memory
Undefined line number
Subscript out of range
Duplicate Definition
Division by zero
Illegal direct

Type mismatch
Out of string space
String too long
String formula too complex
Can't continue
Undefined user function
No RESUME
RESUME without error
Unprintable error
Missing operand
Line buffer overflow
FOR without NEXT
WHILE without WEND
WEND without WHILE
Graphics statement not implemented

□ BASIC-PLUS

Cuvinte rezervate

ABS, ACS, ASCII, AND, AS, ASN, APPEND, ATN

CHAIN, CHR, CLOSE, COMMON, CVT CONTINUE, COS, COMP %, COUNT

DATA, DATE$, DEF, DELETE, DIM, DIF$, DROUND

ELSE, END, EXP, ERROR, EQV

FIELD, FILE, FILE SIZE, FIX, FN, FNEND, FOR, FRACT

GET, GOSUB, GOT

HELP

INT, IF, INPUT, INSTR, IMP

KILL

QUOS

LEFT, LET, LEN, LENGTH, LINE, LIST, LOAD, LOG, LOG10, LSET

MAT, MAT INPUT, MAT PRINT, MAT READ, MAX, MID, MIN, MOD, MODIFY

NEXT, NEW, NOLIST, NRE, NUM$

ON, OPEN, OR, OUTPUT

PLACE$, PI, PRINT, PROD$, PUT

RAD, RANDOM, RANDOMIZE, READ, RECORD, REMARK, RENAME, REPLACE, RESTORE, RESUME, RETURN, REV$, RIGHT, RSET, RND, RUN

SAVE, SGN, SEQUENTIAL, SIN, SQR, SPACE, STEP, STRING$, SUM$, STOP

TAB, TAN, THEN, TIME$, TO, TRANSLATE

UNSAVE, USING, UNTIL, UNLESS

VAL, VIRTUAL

XLATE, XOR

WHILE, WRITE

Mesaje de erori

SYNTAX AND SEMANTIC ERRORS TABLE

1 : * ILLEGAL CONDITIONAL EXPRESSION *
2 : * ILLEGAL DUMMY VARIABLE IN A USER-DEFINED FUNCTION *
3 : * ILLEGAL EXPRESSION *
4 : * PRINT-USING FORMAT ERROR *
5 : * ILLEGAL FUNCTION NAME *
6 : * IF STATEMENT SYNTAX ERROR *
7 : * COMMAND ? VERB ? DON'T UNDERSTAND! *
8 : * ILLEGAL LINE NUMBER (REAL OR GREATER THAN 32767) *
9 : * INVALID NUMERIC OR STRING CONSTANT *
10 : * ILLEGAL BASIC VERB OR MISSING SPECIAL FEATURE *
11 : * NUMERIC PARAMETER IN NEEDED *
12 : * GOTO OR GOSUB IS NECESSARY *
13 : * ILLEGAL COMMON VARIABLE REDEFINITION *
14 : * INSTRUCTION SYNTAX ERROR *
15 : * ILLEGAL IN IMMEDIATE MODE *
16 : * SYNTAX ERROR IN DIM STATEMENT *

17 : * MORE THAN TWO DIMENSIONS FOR MATRIX *
18 : * SYNTAX ERROR IN COMMON STATEMENT *
19 : * BASIC LINE SYNTAX ERROR *
20 : * ILLEGAL FOR THIS MOMENT *
21 : * MAT–INVERT MATRIX FOR INTEGER DESTINATION *
22 : * FLOATING POINT SYNTAX ERROR *
23 : * MISSING SPECIAL FUNCTION OR FEATURE *
24 : * INTEGER NUMBER TOO BIG *
25 : * ERROR IN STRING EXPRESSION *
26 : * ILLEGAL VERB IN IF STATEMENT *
27 : * INTEGER SYNTAX ERROR *
28 : * SYNTAX ERROR IN MAT EXPRESSION *
29 : * SYNTAX ERROR IN MAT INSTRUCTION *
30 : * ILLEGAL I–O CHANNEL NAME *
31 : * ILLEGAL MATRIX REDEFINITION *
32 : * RECORD–ERROR IN RECORD NUMBER *
33 : * PRINT–MISSING COMMA OR SEMICOLON AS SEPARATOR *
34 : * FILESIZE–ERROR IN FILE SIZE EXPRESSION *
35 : * TAB–BAD INDEX LINE EXPRESSION *
36 : * RECORDSIZE–ERROR IN RECORD SIZE EXPRESSION *
37 : * FILE SPECIFICATION ERROR *
38 : * CAN'T CONTINUE, EXECUTION CANNOT BE RESUMED *
39 : * ILLEGAL PROGRAM NAME *
40 : * MAXIMUM CORE EXCEEDED; PLEASE USE SAVE, LOAD COMMANDS *
41 : * OPEN–PARAMETER SYNTAX ERROR *
42 : * NO PROGRAM IN MEMORY *
43 : * BAD LINE NUMBER PAIR IN LIST, DELETE OR RUN COMMAND *
44 : * #NO LIST# PARAMETER IN ACTION; CAN'T LIST OR SAVE *

USER RECOVERABLE ERRORS TABLE

129 : * OVERFLOW IN FLOATING POINT OPERATION *
130 : * UNDERFLOW IN FLOATING POINT OPERATION *
131 : * INTEGER BINARY OVERFLOW *
132 : * NO DATA INSTRUCTION *
133 : * STRING SIZE OVERFLOW *
134 : * ILLEGAL DATA ITEM OR WRONG VARIABLE TYPE *
135 : * NOT ENOUGH ITEMS IN DATA BLOCK *
136 : * END OF FILE MARK FOUNDED *

NON-RECOVERABLE ERRORS TABLE

193 : * NO PROGRAM PRESENT FOR RUN, SAVE, REPLACE *
194 : * UNDEFINED DESTINATION FOR A TRANSFER OF CONTROL *
195 : * NOT ENOUGH MEMORY FOR PROGRAM OR COMMAND EXECUTION *
196 : * FOR ... NEXT NESTED INCORRECTLY OR MORE THAN 12 *
197 : * INVALID USER FUNCTION CALL *
198 : * FOR ... NEXT LOOP INCOMPLETE, MISSING CORRESPONDING NEXT *
199 : * RESTORE USED FOR INVALID OR UNDEFINED BLOCK DATA *
200 : * SUBSCRIPT OUT OF DEFINED RANGE *
201 : * UNDEFINED FUNCTION CALL *
202 : * READ USED WITHOUT DATA *
203 : * FILE READ ERROR; CORRUPTED FILE STRUCTURE *
204 : * FILE WRITE ERROR; CORRUPTED FILE STRUCTURE *
205 : * FILE CLOSE ERROR; CORRUPTED FILE STRUCTURE *
206 : * FILE DOES NOT EXIST *
207 : * FILE OPEN ERROR; CORRUPTED FILE STRUCTURE *
208 : * FILE KILL ERROR; CORRUPTED FILE STRUCTURE *
209 : * BAD SEQUENCE *
210 : * FIELD–BLOCK BUFFER OVERFLOW *
211 : * ERROR IN LOG FUNCTION, ARGUMENT ZERO OR NEGATIVE *
212 : * ERROR IN EXP FUNCTION, ARGUMENT TOO LARGE *
213 : * ERROR IN EXP FUNCTION, ARGUMENT TOO SMALL *
214 : * ERROR IN SQR FUNCTION, NEGATIVE ARGUMENT *

215 : * ERROR IN R∧R FUNCTION, NEGATIVE BASE POWER *
216 : * LEFT FUNCTION, INVALID ARGUMENT *
217 : * RIGHT FUNCTION, INVALID ARGUMENT *
218 : * MID FUNCTION, INVALID ARGUMENT *
219 : * CHR$ OR ASCII FUNCTION, INVALID ARGUMENT *
220 : * INSTR FUNCTION, INVALID ARGUMENT *
221 : * STRING$ FUNCTION, INVALID ARGUMENT *
222 : * INPUT–OUTPUT CHANNEL ALREADY USED *
223 : * INVALID FILE SPECIFICATION *
224 : * INVALID I–O CHANNEL NUMBER, NEGATIVE OR GREATER THAN 8 *
225 : * I–O CHANNEL ALREADY FREE OR UNUSED *
226 : * I–O CHANNEL NOT AVAILABLE *
227 : * ERROR IN I∧I FUNCTION, I=0 AND J=0 OR J<0 *
228 : * ERROR IN R∧I FUNCTION, R=0 AND I=0 OR I<0 *
229 : * VAL FUNCTION, STRING ARGUMENT IS NEEDED *
230 : * MAT – REDIMENSIONED MATRIX TOO LARGE *
231 : * SPACE$ FUNCTION, INVALID ARGUMENT *
232 : * IDN FUNCTION, MATRIX NOT SQUARE *
233 : * CAN'T CREATE NEW FILE *
234 : * FILE ALREADY EXIST, ILLEGAL DUPLICATE FILE *
235 : * INPUT – INVALID DATA TYPE, RETYPE IT *
236 : * OPERAND MATRICES NOT CONFORMABLE DIMENSIONED *
237 : * IMPOSSIBLE TO CHAIN PROGRAMS, PROGRAM LOST. SORRY ! *
238 : * #NOLIST# ACTIF; CAN'T LIST OR SAVE SOURCE PROGRAM *
239 : * CAN'T INVERT MATRIX; NEARLY SINGULAR MATRIX *
240 : * EXECUTED OPEN NOT PERMITS THIS KIND OF OPERATION *
241 : * ILLEGAL FUNCTION REDEFINITION *
242 : * ILLEGAL MATRIX REDEFINITION *
243 : * INVALID OR ILLEGAL OPTION *
244 : * MATRIX NOT DIMENSIONED OR ILLEGAL REDIMENSION *
245 : * LOGICAL ERROR OR FNEND NOT IN PROPER PLACE *
246 : * ODD, NON-EXISTENT ADDRES. CATASTROPHIC ERROR. TYPE NEW ! *
247 : * MEMORY PROTECT VIOLATION. CATASTROPHIC ERROR. TYPE NEW ! *
248 : * RESERVED INSTRUCTION. CATASTROPHIC ERROR. TYPE NEW ! *
249 : * APPEND COMMAND. ILLEGAL. BIC TYPE PROGRAM *
250 : * DEVICE NOT FILE STRUCTURED FOR VIRTUAL OR RANDOM FILE *
251 : * MAT FUNCTION NOT AVAILABLE FOR THIS CONFIGURATION *
252 : * ERROR DURING USER ERROR SUBROUTINE EXECUTION *
253 : * VIRTUAL FILE NOT YET OPEN *
254 : * ASN OR ACS FUNCTION ARGUMENT NOT IN RANGE *

□ ABASIC

Cuvinte rezervate

ABS (valoare absolută), **AND** (operator logic ŞI), **ASC** (furnizare cod ASCII), **ATN** (funcţia arctangentă), **AUTO** (inserare linii)

BASE (modifică adresarea primului element al unui tablou), **BEL** (generare şir X'2F' «alarmă sonoră»)

CHAIN (înlănţuire programe), **CHR** (conversie cod ASCII în cod EBCDIC), **CHRA** (conversie cod ASCII în cod EBCDIC), **CHRE** (conversie număr în cod EBCDIC), **CINT** (cel mai apropiat întreg), **CLEAR** (iniţializare variabile), **CLOSE** (închidere fişier(e)), **CON** (reluare execuţie), **CONT** (reluare execuţie), **COS** (funcţia cosinus), **CPOS** (furnizare poziţie curentă în linia de ieşire)

DATA (generare constante), **DATE** (furnizare dată curentă), **DAY** (furnizare dată curentă), **DEF** (definire funcţie utilizator), **DEFDBL** (definire tip implicit dublă precizie), **DEFFN** (definire funcţie utilizator), **DEFINT** (definire tip implicit întreg), **DEFSTR** (definire tip implicit şir), **DELETE** (ştergere linii), **DIGIT** (modifică număr zecimal de afişat), **DIM** (dimensionare matrici), **DIMENSION** (dimensionare matrici), **DIV** (operator de împărţire întreagă)

EBC (furnizare cod EBCDIC), **EDIT** (modificare linie), **ELSE** (alternativă execuţie condiţionată), **END** (terminare execuţie), **ENDIF** (sfîrşit execuţie condiţionată), **ENDPROC** (sfîrşit definire procedură), **ENDWHILE** (sfîrşit execuţie repetitivă pe condiţie), **EOF** (furnizare −1 pe sfîrşit fişier), **EQV** (operator logic de echivalenţă), **ERL** (furnizare număr ultima linie cu eroare), **ERR** (furnizare cod ultima eroare), **ERROR** (simulare apariţie eroare), **EXEC** (execuţie procedură), **EXIT** (trecere în stare EDITOR), **EXP** (funcţia exponenţială).

FIX (funcţia parte trunchiată), **FN** (definire funcţie utilizator), **FOR** (execuţie repetitivă cu contor), **FRE** (furnizare număr octeţi disponibili), **FREE** (furnizare număr octeţi disponibili)

GO (salt), **GOTO** (salt), **GO TO** (salt), **GOSUB** (salt la subrutină cu revenire)

HEX (reprezentare hexazecimală argument)

IF (execuţie condiţionată), **IMAGE** (furnizare format de editare), **IMP** (operator logic implicaţie), **INPUT** (intrare date de la terminal), **INPUT** (intrare date din fişier), **INSTR** (furnizare poziţie subşir într-un şir), **INT** (funcţia parte întreagă)

KILL (ştergere program (fişier ARIEL))

LEFT (extragere subşir marginea stîngă), **LEN** (furnizare lungime şir), **LET** (atribuire), **LFT** (extragere subşir margine stîngă), **LG** (logaritm zecimal), **LINE INPUT** (intrare date de la terminal), **LINE INPUT** (intrare date din fişier), **LINPUT** (intrare date de la terminal), **LINPUT** (intrare date din fişier), **LIST** (listare program), **LN** (logaritm natural), **LOAD** (încărcare program (fişier ARIEL)), **LOG** (logaritm natural), **LOG10** (logaritm zecimal)

MERGE (interclasare programe şi execuţie program obţinut), **MID** (extragere subşir de la o poziţie dată pe o lungime dată), **MOD** (operator modulo)

NEW (ştergere program încărcat), **NEXT** (incrementare contor execuţie repetitivă), **NOT** (operator logic negaţie)

OCT (reprezentare octală argument), **ON** (salt condiţionat), **ON** (salt condiţionat la subrutină), **ON ERROR** (salt la subrutină pe condiţie de eroare), **OPEN** (deschidere fişiere), **OPTION** (BASE, WIDTH, DIGIT), **OR** (operator logic SAU)

PI (numărul PI), **POS** (poziţie subşir într-un şir), **PRINT** (ieşire date pe terminal), **PRINT** (scriere date în fişier), **PRINT USING** (ieşire date cu format editare), **PRINT USING** (ieşire date cu format editare pe fişier), **PROC** (definire procedură)

RANDOMIZE (modificare bază generator de numere aleatoare), **READ** (intrare date din zona de memorie (**DATA**)), **REM** (comentariu), **RENUM** (re-

numerotare linii program), **RESTORE** (repoziţionare pe început într-un bloc **DATA**), **RESUME** (terminare subrutină tratare erori), **RETURN** (revenire din subrutină), **RGT** (extragere subşir margine dreaptă), **RIGHT** (extragere subşir margine dreaptă), **RND** (generare numere aleatoare), **RUN** (lansare în execuţie)

SAVE (salvare program (fişier ARIEL)), **SEG** (extragere subşir de la poziţia ... la poziţia), **SGN** (furnizare semn expresie), **SIGN** (furnizare semn expresie), **SIN** (funcţia sinus), **SPACE** (generare şir de blancuri), **SPC** (generare şir de blancuri), **SQR** (funcţia radical), **ŞTEP** (pasul conturului de repetiţie), **STOP** (oprire execuţie), **STR** (conversie număr în şir (reprezentarea acestuia)), **STRING** (creare şir de lungime dată), **SRG** (creare şir de lungime dată), **SWAP** (interschimbare valori variabile)

TAB (setare poziţie curentă în linie ieşire), **TAN** (funcţia tangentă), **TG** (funcţia tangentă), **THEN** (alternativă la execuţie condiţionată), **TIME** (furnizare ora curentă), **TO** (salt), **TO** (valoare finală execuţie repetitivă (**FOR**)), **TRM** (furnizare şir echivalent fără blancuri la sfîrşit)

UNTIL (condiţie de execuţie repetitivă), **USING** (descriere format editare)

VAL (conversie şir – reprezentare număr – în valoare numerică)

WEND (sfîrşit bloc **WHILE**), **WHILE** (execuţie repetitivă pe condiţie), **WIDTH** (definire parametri linie de editare), **WRITE** (ieşire date pe terminal)

XCHANGE (renumerotare linii program), **XOR** (operator logic SAU-EXCLUSIV)

Mesaje de erori

Cod	Mesaj	Cod	Mesaj
1	NO ROOM ON VIRTUAL FILE	2	USER'S ROOM EXCEEDED
10	SYNTAX	11	DATA OVERFLOW
12	DATA UNDERFLOW	13	ARGUMENTS NUMBER?
14	ARGUMENT TYPE?	15	NEGATIVE ARGUMENT
16	LINE UNKNOWN	17	CONTINUE NOT ALLOWED
18	MATRIX	19	ELSE WITHOUT IF
20	STRING LENGTH > 140	21	FOR WITHOUT NEXT
22	IF BLOCK WITHOUT ENDIF	23	RETURN WITHOUT GOSUB
24	STACK OVERFLOW	25	INCOHERENT ARGUMENTS
26	WHILE WITHOUT WEND	27	WEND WITHOUT WHILE
28	ENDIF WITHOUT IF	29	TOO MANY NESTED FOR-NEXT
30	NESTING OVERFLOW	31	FUNCTION
32	NO DATA DURING READ	33	PROC WITHOUT ENDPROC
34	PROGRAM UNKNOWN	35	ILLEGIBLE PROGRAM
36	LINE NUMBER OVERFLOW	37	FILE NAME
38	ZPA FULL	39	I/0 DISK
40	FILE EXISTS	41	FILE IS PUBLIC
42	FILE UNKNOWN	43	FILE IN USED
44	OPEN ACTIVE FILE	45	WHERE IS OPEN?
46	WRITE ON INPUT FILE	47	READ ON OUTPUT FILE
48	MORE THAN 8 ACTIVE FILES	49	FILE NUMBER
50	CLOSE INACTIVE FILE	51	INCONGROUS DATA FOR INPUT#
52	INPUT# PAST EOF		



SINTEZA 16

Mai mult decît un memento al limbajului BASIC-AMSTRAD

□ Comenzi și instrucțiuni BASIC-AMSTRAD

ABS

ABS (⟨expresie numerică⟩)
PRINT ABS (–67·98)
67·98

Funcțiune. Calculează valoarea **ABS**olută a expresiei dintre paranteze.
Cuvinte cheie asociate. **SGN**

AFTER

AFTER ⟨durata cronometrului⟩ [, ⟨număr de cronometru⟩] **GOSUB** ⟨număr de linie⟩

```
10 AFTER 250 GOSUB 60 : CLS
20 PRINT "Ghicește o literă în cinci secunde"
30 a$=INKEY$ : IF flag=1 THEN END
40 IF a$ <> CHR$ (INT(RND * 26+97)) THEN 30
50 PRINT a$; "este exact, ai cîștigat!"
55 SOUND 1, 478 : SOUND 1, 358 : END
60 PRINT "Prea tîrziu. Eu am cîștigat!"
70 SOUND 1, 2000 : flag=1 : RETURN
```

Comandă. Apelează un subprogram după (**AFTER**, în engleză) un anumit timp. Timpul (durata) cronometrului indică durata așteptării în multipli de 0,02 secunde. ⟨Număr de cronometru⟩ (care poate fi 0, 1, 2 sau 3) precizează pe care din cele patru cronometre de așteptare trebuie să-l utilizăm.

Fiecare din cele patru cronometre se poate asocia unui subprogram.
Cuvinte cheie asociate. **EVERY, REMAIN, RETURN**

AND

⟨argument⟩ **AND** ⟨argument⟩
IF "alin" ⟨"bebe" **AND** "cîine"⟩ "pisică" **THEN PRINT** "adevărat" **ELSE PRINT** "fals"
adevărat

Operator. Execută operaţii booleene (ŞI).
Cuvinte cheie asociate. **OR, NOT, XOR**

ASC

ASC (⟨şir de caractere alfanumerice⟩)
PRINT ASC("x")
120

Funcţiune. Calculează valoarea numerică a primului caracter dintr-un şir de caractere.
Cuvinte cheie asociate. **CHR$**

ATN

ATN (⟨expresie numerică⟩)
PRINT ATN(1)
0.785398163

Funcţiune. Calculează Arcul TaNgent (reducînd expresia numerică la un număr real în radiani, cuprins între –PI/2 şi +PI/2) al valorii date.

Observaţie. Comenzile **DEG** şi **RAD** pot fi utilizate pentru a specifica explicit că rezultatul se va exprima respectiv în grade sau radiani.

Cuvinte cheie asociate. **COS, DEG, RAD, SIN, TAN**

AUTO

AUTO [⟨număr de linie⟩] [,⟨increment⟩]
AUTO 100, 50

Comandă. Generează **AUTO**mat numerele de linie. Parametrul facultativ ⟨număr de linie⟩ dă primul număr de linie care trebuie generat. Dacă nu îl precizaţi, liniile se vor genera începînd cu 10.

Incrementul, tot facultativ, stabileşte intervalul dintre numerele de linie. În lipsa specificaţiei, el va fi egal cu 10. Dacă se generează un număr de linie deja utilizat, pe ecran apare conţinutul acestei linii şi poate fi, eventual, modificat. Linia afişată este apoi înlocuită în memorie după activarea tastei **[RETURN]**. Pentru a opri numerotarea automată a liniilor, tastaţi **[ESC]**.

Cuvinte cheie asociate. Nu există.

BIN$

BIN$ (⟨număr întreg fără semn⟩ [,⟨număr întreg⟩])
PRINT BIN$ (64, 8)
01000000

Funcţiune. Produce un şir de cifre BINare ce reprezintă valoarea ⟨numărului întreg fără semn⟩, cu ajutorul numărului de cifre binare indicat de cel de-al doilea ⟨număr întreg⟩ (între 0 şi 16). Dacă acest număr este prea

mare, rezultatul începe cu atîtea zero-uri cîte sînt necesare. Dacă el este prea mic, rezultatul nu este trunchiat, ci convertit în atîtea cifre cît este necesar.

⟨Numărul întreg fără semn⟩ care trebuie convertit în număr binar trebuie să fie cuprins între –32768 şi 65535.

Cuvinte cheie asociate. **DEC$, HEX$, STR$**

BORDER

BORDER ⟨număr de culoare⟩ [,⟨număr de culoare⟩]

```
10 REM 729 combinaţii
20 SPEED INK 5,5
30 FOR a=0 TO 26
40 FOR b=0 TO 26
50 BORDER a, b : CLS : LOCATE 14, 13
60 PRINT "margine"; a; " , "; b
70 FOR t=1 TO 500
80 NEXT t, b, a
```

Comandă. Pentru schimbarea culorii marginii ecranului. Dacă se indică două culori, ele alternează cu o viteză determinată prin comanda **SPEED INK.** Valorile sînt cuprinse între 0 şi 26.

Cuvinte cheie asociate. **SPEED INK**

BREAK

(A se vedea **ON BREAK CONT, ON BREAK GOSUB, ON BREAK STOP)**

CALL

CALL ⟨adresă⟩ [,⟨listă parametri⟩]

```
CALL 0
```

Comandă. Permite unui subprogram extern să fie apelat pornind din BASIC. Exemplul de mai sus reiniţializează complet calculatorul.

De utilizat cu multă atenţie.

Cuvinte cheie asociate. **UNT**

CAT

CAT

```
CAT
```

Comandă. Cere BASIC-ului să citească **CAT**alogul dischetei. Afişează în ordine alfanumerică numele tuturor fişierelor prezente, precum şi lungimea lor (întregind Koctetul superior). Numărul de octeţi disponibili este, de asemenea, afişat, cu identificarea dischetei şi a utilizatorului.

Această comandă nu are influenţă asupra programului în curs.

Cuvinte cheie asociate. **LOAD, RUN, SAVE**

CHAIN

CHAIN ⟨nume fişier⟩ [,⟨număr de linie⟩]
CHAIN "testprog. bas", 350

Comandă. Încarcă un program în memorie pornind de la o dischetă, înlocuind programul existent.

Fişierele protejate (salvate prin comanda **SAVE,** p) se pot încărca şi lansa prin **CHAIN.**

Cuvinte cheie asociate. **CHAIN MERGE, LOAD, MERGE**

CHAIN MERGE

CHAIN MERGE ⟨nume fişier⟩ [,⟨număr de linie⟩]
[,**DELETE** ⟨ansamblu de linii⟩]
CHAIN MERGE "pàrtea 2. jos", 750, **DELETE** 400–680

Comandă. Încarcă în memorie un program de pe dischetă, contopindu-l cu programul existent, apoi lansează programul care rezultă.

Cuvinte cheie asociate. **LOAD, MERGE, DELETE, CHAIN**

CHR$

CHR$ (⟨număr întreg⟩)

```
10 FOR x=32 TO 255
20    PRINT x; CHR$(x),
30 NEXT
```

Funcţiune. Transformă un ⟨număr întreg⟩ cuprins între 0 şi 255 într-un şir de caractere echivalent. Caracterele de la 0 la 31 sînt caractere de control.

Cuvinte cheie asociate. **ASC**

CINT

CINT (⟨expresie numerică⟩)

```
10 n=1.9999
20 PRINT CINT(n)
run
2
```

Funcţiune. Transformă o valoare numerică într-un număr întreg rotunjit cuprins între –32768 şi 32767.

Cuvinte cheie asociate. **CREAL, FIX, INT, ROUND, UNT**

CLEAR

CLEAR
CLEAR

Comandă. Şterge toate variabilele, fişierele deschise, tabelele şi funcţiile utilizator.

Cuvinte cheie asociate. Nu există.

CLEAR INPUT

CLEAR INPUT
10 **CLS**
20 **PRINT** "tastaţi mai multe litere!"
30 **FOR** t=1 **TO** 3000
40 **NEXT**
50 **CLEAR INPUT**

Comandă. Şterge toate datele introduse de la claviatură, care se găsesc in buffer.

Cuvinte cheie asociate. **INKEY, INKEY$, JOY**

CLG

CLG [⟨cerneală⟩]
LOCATE 1, 20
CLG 3

Comandă. Şterge ecranul grafic şi-i redă culoarea de fond.

Cuvinte cheie asociate. **CLS, GRAPHICS PAPER, INK, ORIGIN**

CLOSEIN

CLOSEIN
CLOSEIN

Comandă. Închide orice fişier de intrare deschis pe dischetă (a se vedea **OPENIN**).

Cuvinte cheie asociate. **EOF, OPENIN**

CLOSEOUT

CLOSEOUT
CLOSEOUT

Comandă. Închide orice fişier de ieşire deschis (a se vedea **OPENOUT**).

Cuvinte cheie asociate. **OPENOUT**

CLS

CLS *[#⟨număr de canal⟩]*
10 **PAPER** #2, 3
20 **CLS** #2

Comandă. Şterge fereastra de pe ecran specificată prin ⟨număr de canal⟩ şi îi dă culoarea sa de "hîrtie". În lipsa ⟨numărului de canal⟩, ia 0.

Cuvinte cheie asociate. **CLG, INK, PAPER, WINDOW**

CONT

CONT
CONT

Comandă. **CONT**inuă execuţia programului după **STOP,** sau două activări ale tastei **[ESC],** dacă programul nu a fost nici modificat şi nici protejat. Se pot tasta comenzi directe înainte de a relua programul.
Cuvinte cheie asociate. **STOP**

COPY CHR$

COPY CHR$ (#⟨număr de canal⟩)

```
10 CLS
20 PRINT "colţ superior"
30 LOCATE 1, 1
40 a$=COPY CHR$ (0)
50 LOCATE 1, 20
60 PRINT a$
```

Funcţiune. Copiază un caracter pornind de la poziţia cursorului (care TREBUIE să fie specificată). Programul de mai sus copiază un caracter din poziţia 1,1 (colţ superior stînga) şi îl reproduce în 1,20.
Cuvinte cheie asociate. **LOCATE**

COS

COS (⟨expresie numerică⟩)

```
DEG
PRINT COS (45)
0.707106781
```

Funcţiune. Calculează **COS**inusul ⟨expresiei numerice⟩. **DEG** şi **RAD** pot servi la exprimarea argumentului în grade sau radiani.
Cuvinte cheie asociate. **ATN, DEG, RAD, SIN**

CREAL

CREAL (⟨expresie numerică⟩)

```
10 a=PI
20 PRINT CINT(a)
30 PRINT CREAL(a)
run
3
3.14159265
```

Funcţiune. Transformă ⟨expresie numerică⟩ în număr real.
Cuvinte cheie asociate. **CINT**

CURSOR

```
CURSOR [(indicator sistem)] [,(indicator utilizator)]
  10 CURSOR 1
  20 PRINT "întrebare?";
  30 a$=INKEY$ : IF a$="" THEN 30
  40 PRINT a$
  50 CURSOR 0
```

Comandă. Activează sau dezactivează indicatorul sistem sau utilizator.
Cuvinte cheie asociate. **LOCATE**

DATA

```
DATA (listă de constante)
  10 FOR x=1 TO 4
  20    READ nume$, prenume$
  30    PRINT "Tov."; nume$; " "; prenume$
  40 NEXT
  50 DATA DAMIAN, Olimpiu, DINU, Ion
  60 DATA LASCU, Vasile, MANIU, Daniel
```

Comandă. Declară constante în interiorul programului.
Cuvinte cheie asociate. **READ, RESTORE**

DEC$

```
DEC$ ((expresie numerică), (model de format))
   PRINT DEC$ (107, "££##########,·##")
  £10,000,000.00
```

Funcţiune. Dă o reprezentare **DEC**imală a expresiei numerice, utilizînd (modelul de format) indicat.

Modelul de format nu poate să conţină DECÎT caracterele:
\+ – £ $ * # , . ↑

Folosirea acestor "indicatori de format" este descrisă la cuvîntul cheie **PRINT USING.**

Cuvinte cheie asociate. **BIN$, HEX$, PRINT USING, STR$**

DEF FN

```
DEF FN (nume) [((parametri formali))]=(expresie)
  10 t=TIME/300
  20 DEF FNchrono=INT (TIME/300-t)
```

Comandă. Defineşte o funcţie scrisă de utilizator.
Cuvinte cheie asociate. Nu există.

DEFINT

```
DEFINT (listă de litere)
  10 DEFINT n
```

```
20 număr=123·456
30 PRINT număr
run
123
```

Comandă. Această comandă definește tipul variabilelor după prima literă a numelui variabilei. Ea poate fi urmată de o listă de inițiale. De exemplu:

DEFINT a, b, c

sau:

DEFINT a–z

Cuvinte cheie asociate. **DEFREAL, DEFSTR**

DEFREAL

DEFREAL ⟨listă de litere⟩
    DEFREAL· x, a–f

Comandă. Definește tipul de variabilă.
Tipul variabilei va fi determinat după prima literă a numelui variabilei. Ea poate fi urmată de o listă de inițiale.

DEFREAL a, b, c

sau:

DEFREAL a–z

Cuvinte cheie asociate. **DEFINIT, DEFSTR**

DEFSTR

DEFSTR ⟨listă de litere⟩

```
10 DEFSTR n
20 nume="Amstrad"
30 PRINT nume
run
Amstrad
```

Comandă. Tipul de variabilă este determinat după prima literă din numele său. Comanda poate fi urmată de o listă de inițiale:

DEFSTR, a, b, c

sau:

DEFSTR a–z

Cuvinte cheie asociate. **DEFINT, DEFREAL**

DEG

DEG
    DEG

Comandă. Stabilește modul de calcul în **DEG**rade (grade). În schimb, funcțiile **SIN, COS, TAN** și **ATN** consideră că argumentul care li se trans-

mite este exprimat în radiani. Comanda rămîne valabilă pînă cînd se utilizează comenzile **RAD** sau **NEW, CLEAR, LOAD, RUN** etc.

Cuvinte cheie asociate. **ATN, COS, RAD, SIN, TAN**

DELETE

DELETE ⟨ansamblu de linii⟩
  DELETE 100–200

Comandă. Șterge o parte a programului definit în ⟨ansamblul de linii⟩.

Cuvinte cheie asociate. **CHAIN MERGE, RENUM**

DERR

DERR
  LOAD "xyz·abc"
  XYZ·ABC not found
  Ready
  PRINT DERR
  146

Funcțiune. Raportează ultimul cod de eroare transmis de sistemul de gestiune al dischetei. Valoarea lui **DERR** poate servi la confirmarea erorii descoperite. Consultați lista de mesaje de erori.

Cuvinte cheie asociate. **ERL, ERR, ERROR, ON ERROR GOTO, RESUME**

DI

DI

Comandă. Dezactivează o întrerupere (alta decît [**ESC**]) pînă cînd ea este reactivată direct printr-o comandă **EI** sau indirect printr-un [**RETURN**] la sfîrșitul unui subprogram de întrerupere **GOSUB**.

Intrarea într-un subprogram de întrerupere dezactivează automat întreruperile cu prioritate egală sau mai mică.

Este utilizată atunci cînd programul trebuie să se execute fără întrerupere, de exemplu atunci cînd sînt în competiție două subprograme pentru a folosi resursele calculatorului (resurse grafice, de exemplu).

Cuvinte cheie asociate. **AFTER, EI, EVERY, REMAIN**

DIM

DIM ⟨listă de variabile indexate⟩

Comandă. **DIM**ensionează un tablou. Această comandă alocă spațiul necesar tablourilor și specifică valorile maxime de indici. BASIC-ul trebuie să cunoască spațiul rezervat pentru un tablou, în lipsa specificației, el ia valoarea 0.

Un tablou se identifică printr-o variabilă indexată, şi anume un nume de variabilă însoţit de un ansamblu de indici, astfel ca fiecare „element" al tabloului să aibă propria sa valoare de indice.

O buclă **FOR NEXT** poate servi la controlul tabloului, prelucrînd fiecare element al tabloului pe rînd.

Valoarea minimă a unui indice este zero (acesta este primul element dintr-un tablou).

Tablourile pot fi multi-dimensionale şi fiecare element este referit prin poziţia sa. De exemplu, într-un tablou dimensionat prin:

DIM poziţie (20, 20, 20)

... un element al tabloului va fi referit în felul următor:

poziţie (4, 5, 6)

Cuvinte cheie asociate. **ERASE**

DRAW

DRAW ⟨coordonată x⟩, ⟨coordonată y⟩, [,[⟨cerneală⟩] [,⟨tip de cerneală⟩]]

```
10 MODE 0 : BORDER 0 : PAPER 0 : INK 0,0
20 x=RND * 640 : y=RND * 400 : z=RND * 15
30 DRAW x,y,z
40 GOTO 20
```

Comandă. Trasează o linie pe ecran între poziţia cursorului (poziţie grafică) şi o poziţie absolută specificată de coordonatele x şi y. ⟨Cerneala⟩ de trasaj poate fi specificată (între 0 şi 15).

⟨tip de cerneală⟩ (facultativ) determină interacţiunea cernelii pe afişajul prezent pe ecran. Cele patru ⟨tipuri de cerneală⟩ sînt următoarele:

0 : Normală
1 : **XOR** (SAU exclusiv)
2 : **AND** (ŞI)
3 : **OR** (SAU)

Cuvinte cheie asociate. **DRAWR, GRAPHICS, PEN, MASK**

DRAWR

DRAWR ⟨deplasare x⟩, ⟨deplasare y⟩ [,[⟨cerneală⟩] [,⟨tip de cerneală⟩]

```
10 CLS : PRINT "urci la primul etaj!?"
20 MOVE 0,350 : FOR n=1 TO 8
30 DRAWR 50,0
40 DRAWR 0, -50
50 NEXT : MOVE 348,0 : FILL 3
60 GOTO 60
```

Comandă. Trasează o linie pe ecranul grafic pornind de la cursorul grafic şi pînă la poziţia specificată de ⟨x şi y⟩. Cerneala de trasaj poate fi specificată (între 0 şi 15) ⟨tip de cerneală⟩ (facultativ) determină interac-

ţiunea cernelii pe afişajul prezent pe ecran. Cele 4 ⟨tipuri de cerneală⟩ sînt următoarele:

0 : Normal
1 : **XOR** (SAU exclusiv)
2 : **AND** (ŞI)
3 : **OR** (SAU)

Cuvinte cheie asociate. **DRAW, GRAPHICS PEN, MASK**

EDIT

EDIT ⟨număr de linie⟩
EDIT 20

Comandă. Afişează linia programului precum şi cursorul pregătit de editare.

Cuvinte cheie asociate. **AUTO, LIST**

EI

EI
EI

Comandă. Activează o întrerupere dezactivată de **DI**.

Întreruperile dezactivate printr-un subprogram de întrerupere sînt automat restabilite prin comanda **RETURN**, la sfîrşitul subprogramului.

Cuvinte cheie asociate. **AFTER, DI, EVERY, REMAIN**

ELSE

(vezi **IF**)

END

END
END

Comandă. Termină execuţia programului şi restabileşte modul direct. Un program poate să conţină un număr oarecare de comenzi **END** (este implicit la sfîrşitul oricărui program BASIC-AMSTRAD)

Cuvinte cheie asociate. **STOP**

ENT

ENT ⟨număr de anvelopă⟩ [, ⟨secţiune de anvelopă⟩][, ⟨secţiune de anvelopă⟩]
[, ⟨secţiune de anvelopă⟩][, ⟨secţiune de anvelopă⟩]
[, ⟨secţiune de anvelopă⟩]

10 **ENT** 1, 10, –50, 10, 10, 50, 10
20 **SOUND** 1, 500, 200, 10, 1

Comandă. Defineşte anvelopa de tonalitate specificată prin ⟨număr de anvelopă⟩ (între 1 şi 15) utilizată cu comanda **SOUND**. Dacă ⟨numărul de anvelopă⟩ este negativ (între −1 şi −15) anvelopa se repetă pînă la sfîrşitul perioadei stabilirii sale prin comanda **SOUND**.

Fiecare ⟨secţiune de anvelopă⟩ poate să conţină 2 sau 3 parametri. În cazul a trei parametri, aceştia sînt:
⟨număr de paşi⟩, ⟨amplitudinea pasului⟩, ⟨durata pasului⟩.

Parametrul 1: ⟨număr de paşi⟩
Specifică ⟨numărul de paşi⟩ de variaţie de tonalitate din interiorul secţiunii de anvelopă. De exemplu, într-o secţiune de notă care durează 10 secunde, puteţi fixa 10 paşi de cîte 1 secundă fiecare. În acest caz, ⟨numărul de paşi⟩ va fi 10.

⟨Numărul de paşi⟩ poate să varieze de la 0 la 239.

Parametrul 2: ⟨amplitudinea pasului⟩
Valoarea trebuie să fie cuprinsă între −128 şi +127. Paşii negativi măresc înălţimea notei, cei pozitivi o coboară. Perioada sonoră minimă este 0.

Parametrul 3: ⟨durata pasului⟩
Specifică durata unui pas prin unităţi de 0,01 secunde. Ea poate să varieze de la 0 la 255 (0 are valoarea 256), durata maximă a unui pas este deci de 2,57 secunde. Dacă nu utilizaţi decît doi parametri, aceştia sînt:
⟨perioada sonoră⟩, ⟨durata pasului⟩

Parametrul 1: ⟨perioada sonoră⟩
Dă valoarea nouă a perioadei. (A se vedea parametrul 2 al comenzii **SOUND**).

Parametrul 2: ⟨durata pasului⟩
Specifică durata pasului unic în unităţi de 0,01 secunde. Poate să varieze între 0 şi 255 (0 avînd valoarea 256).

Durata totală a paşilor nu trebuie să depăşească parametrul ⟨durată⟩ al comenzii **SOUND**, deoarece sunetul se termină atunci înainte de a fi traversat totalitatea paşilor. (Restul anvelopei de tonalitate este în acest caz ignorat). De asemenea, dacă ⟨durata⟩ comenzii **SOUND** depăşeşte durata totală a paşilor sunetul se continuă după ce a traversat toţi paşii şi rămîne constant la tonalitatea finală.

Comanda **ENT** poate fi însoţită de 5 ⟨secţiuni de anvelopă⟩ diferite (fiecare alcătuită din 2 sau 3 parametri).

Primul pas dintr-o anvelopă de tonalitate se execută imediat.

Ori de cîte ori o nouă anvelopă este atribuită unui număr determinat de anvelopă, definiţia precedentă dispare.

Un ⟨număr de anvelopă⟩ fără ⟨secţiune de anvelopă⟩ anulează toate specificaţiile precedente.

Cuvinte cheie asociate: **ENV, SOUND**

ENV

ENV ⟨număr de anvelopă⟩ [, ⟨secţiune de anvelopă⟩] [, ⟨secţiune de anvelopă⟩

[,⟨secţiune de anvelopă⟩][,⟨secţiune de anvelopă⟩]
[,⟨secţiune de anvelopă⟩]
[,⟨secţiune de anvelopă⟩]

10 **ENV** 1, 15, –1, 10, 15, 1, 10
20 **SOUND** 1, 200, 300, 15, 1

Comandă. Defineşte anvelopa de volum care corespunde numărului de anvelopă (între 1 şi 15), utilizat cu comanda **SOUND.**

Poate să conţină 2 sau 3 parametri.

Pentru 3 parametri:

⟨număr de paşi⟩, ⟨amplitudinea pasului⟩, ⟨durata pasului⟩.

Parametrul 1: ⟨număr de paşi⟩

Specifică ⟨numărul de paşi⟩ de volum pe care un sunet trebuie să-l traverseze în secţiunea de anvelopă. De exemplu, într-o secţiune de 10 secunde, puteţi fixa 10 paşi cu volumul de o secundă. Parametrul ⟨număr de paşi⟩ este egal cu 10.

Parametrul poate să varieze de la 0 la 127.

Parametrul 2: ⟨amplitudinea pasului⟩

Poate face să varieze volumul de la 0 la 15 în raport cu pasul precedent. Cele 16 volume diferite sînt aceleaşi cu cele din comanda **SOUND.** Parametrul ⟨amplitudinea pasului⟩ poate totuşi să varieze de la –128 la +127, volumul revenind la 0 după ce a atins 15.

Parametrul 3: ⟨durata pasului⟩

Specifică durata unui pas în unităţi de 0,01 secunde. Poate să varieze de la 0 la 255 (0 are valoarea 256).

Pentru 2 parametri:

⟨anvelopă fizică⟩, ⟨perioada anvelopei⟩

Parametrul 1: ⟨anvelopa fizică⟩

Specifică valoarea ce trebuie trimisă la registru de anvelopă cuprins în generatorul sonor.

Parametrul 2: ⟨perioada anvelopei⟩

Specifică valoarea ce trebuie trimisă la registrele de perioadă de anvelopă cuprinse în generatorul sonor.

Utilizarea de anvelope fizice presupune cunoaşterea materialului. În caz contrar, este mai bine să utilizaţi o anvelopă logică integrînd un parametru ⟨durata pasului⟩ adecvat.

Durata totală a paşilor nu trebuie să depăşească parametrul de ⟨durată⟩ al comenzii **SOUND,** sunetul se termină în acest caz înainte de a fi traversat toţi paşii de volum (restul anvelopei se ignoră).

De asemenea, dacă parametrul de ⟨durată⟩ al comenzii **SOUND** depăşeşte durata totală a paşilor, sunetul se continuă după ce a traversat toţi paşii de volum şi rămîne constant la volumul final.

Comanda **ENV** poate să conţină 5 secţiuni de anvelope diferite (alcătuite din cei doi sau trei parametri).

Primul pas al unei anvelope de volum se execută imediat.

Ori de cîte ori o nouă anvelopă este atribuită unui număr de anvelopă determinat, definiţia precedentă dispare.

Specificaţia unui ⟨număr de anvelopă⟩ fără ⟨secţiune de anvelopă⟩ anulează toate valorile precedente.

Cuvinte cheie asociate. **ENT, SOUND**

EOF

EOF

```
10 OPENIN "ex1 . jos"
20 WHILE NOT EOF
30   LINE INPUT#9, a
40   PRINT a
50 WEND
60 CLOSEIN
run
```

Funcţiune. Pentru testarea stării unui fişier deschis. Returnează −1 (adevărat) dacă este la sfîrşitul fişierului (End Of File) sau dacă nici un fişier nu este deschis, dacă nu returnează 0 (fals).

Cuvinte cheie asociate. **OPENIN, CLOSEIN**

ERASE

ERASE ⟨listă de variabile⟩

```
DIM a(100), b(100)
ERASE a, b
```

Comandă. Atunci cînd un tablou nu mai este necesar, el poate să fie şters **(ERASE)** pentru a elibera memoria în scopul altor utilizări.

Cuvinte cheie asociate. **DIM**

ERL

ERL

```
10 ON ERROR GOTO 30
20 GOTO 1000
30 PRINT "Eroarea este în linia"; ERL
40 END
```

Funcţiune. Returnează numărul de linie al ultimei erori întîlnite. În exemplul de mai sus, ERoarea din Linia 20 este indicată prin funcţiunea **ERL.**

Cuvinte cheie asociate. **DERR, ERR, ERROR, ON ERROR GOTO, RESUME**

ERR

ERR

```
GOTO 500
Line does not exist
Ready
PRINT ERR
8
```

Funcţiune. Dă numărul ultimei erori descoperite. În exemplul de mai sus, numărul de eroare 8 corespunde mesajului: „Linia nu există".

Cuvinte cheie asociate. **DERR, ERL, ERROR, ON ERROR GOTO, RESUME**

ERROR

ERROR ⟨număr întreg⟩

```
10 IF INKEY$=" " THEN 10 ELSE ERROR 17
```

Comandă. Stabileşte o acţiune ce urmează unei erori numerotate. Acţiunea este aceeaşi ca cea prevăzută de BASIC în caz de eroare reală, făcînd apel la un subprogram de tratare a erorii.

ERROR însoţit de un ⟨număr întreg⟩ cuprins între 33 şi 255 poate servi la crearea de mesaje de eroare personalizate cum ar fi:

```
10 ON ERROR GOTO 100
20 INPUT "introduceţi un caracter"; a$
30 IF LEN(a$)⟨⟩ 1 THEN ERROR 100
40 GOTO 20
100 IF ERR=100 THEN 110 ELSE 130
110 PRINT CHR$ (7)
120 PRINT "am spus UN caracter!"
130 RESUME 20
```

Cuvinte cheie asociate. **ERL, ERR, ON ERROR GOTO, RESUME**

EVERY

EVERY ⟨durata cronometrului⟩ [, ⟨numărul cronometrului⟩] **GOSUB** ⟨număr de linie⟩

```
10 EVERY 50, 1 GOSUB 30
20 GOTO 20
30 SOUND 1, 20
40 RETURN
run
```

Comandă. Apelează un subprogram din BASIC la intervale regulate. ⟨Durata cronometrului⟩ specifică intervalul prin unităţi de 0,02 secunde. ⟨Numărul de cronometru⟩ (cuprins între 0 şi 3) arată care din cele patru cronometre trebuie utilizat. Cronometrul 3 corespunde priorităţii superioare şi 0 celei inferioare. Fiecare cronometru poate fi asociat unui subprogram.

Cuvinte cheie asociate. **AFTER, REMAIN**

EXP

EXP (⟨expresie numerică⟩)

```
PRINT EXP (6.876)
968.743625
```

Funcţiune. Calculează "e" la puterea dată prin ⟨expresia numerică⟩ în care "e" este egal cu 2,7182818 (aprox.), număr al cărui logaritm natural este 1.

Cuvinte cheie asociate. **LOG**

FILL

FILL ⟨cerneală⟩
```
10 MODE 0
20 FOR n=1 TO 500
30     PRINT "0";
40 NEXT
50 coulstylo=2+RND * 13
60 FILL coulstylo
70 GOTO 50
run
```
Comandă. Completează o zonă la alegere a ecranului grafic. Marginile zonei sînt delimitate de liniile desenate cu cerneala stiloului sau cu cerneala fondului (cuprinsă între 0 şi 15). Completarea porneşte de la poziţia cursorului grafic. Dacă acesta se află pe o margine, nu se completează nimic.

Cuvinte cheie asociate. **GRAPHICS PEN**

FIX

FIX (⟨expresie numerică⟩)
```
PRINT FIX (9.99999)
9
```
Funcţiune. Îndepărtează partea zecimală dintr-o expresie numerică.
Cuvinte cheie asociate. **CINT, INT, ROUND**

FN

(vezi **DEF FN**)

FOR

FOR ⟨variabilă simplă⟩=⟨început⟩**TO**⟨sfîrşit⟩ [**STEP** ⟨increment⟩]
```
10 FOR n=2 TO 8 STEP 2
20    PRINT n;
30 NEXT n
40 PRINT
run
```
Comandă. Execută partea programului care se găseşte între cuvintele cheie **FOR** şi **NEXT,** de atîtea ori de cîte putem adăuga ⟨increment⟩ la ⟨variabila simplă⟩ pornind de la ⟨început⟩ pînă la ⟨sfîrşit⟩. Dacă se omite ⟨increment⟩, el ia implicit valoarea 1.

Valoarea ⟨increment⟩-ului poate fi negativă. În acest caz, valoarea parametrului ⟨început⟩ trebuie să fie superioară valorii parametrului ⟨sfîrşit⟩, în lipsa căreia variabila nu poate fi incrementată.

Afectarea numelui de variabilă la comanda **NEXT** este facultativă deoarece BASIC-ul determină automat comanda **NEXT** căreia i se asociază o comandă **FOR.**

Cuvinte cheie asociate. **NEXT, STEP, TO**

FRAME

```
FRAME
 10 MODE 0
 20 PRINT "fără FRAME"
 30 TAG
 40 MOVE 0, 200
 50 FOR x=0 TO 500 STEP 4
 60    IF f=1 THEN FRAME
 70    MOVE x, 200
 80    PRINT " "; CHR$ (143);
 90 NEXT
100 IF f=1 THEN RUN
110 CLS
120 TAGOFF
130 PRINT "cu FRAME"
140 f=1
150 GOTO 30
run
```

Comandă. Sincronizează scrierea graficelor cu apariţia video. Rezultă o mişcare mai armonioasă a caracterelor sau a graficelor de pe ecran fără distorsiune şi licărire.

Cuvinte cheie asociate. **TAG, TAGOFF**

FRE

FRE (⟨expresie numerică⟩)
FRE (⟨şir de caractere alfanumerice⟩)

```
PRINT FRE (0)
PRINT FRE (" ")
```

Funcţiune. Indică spaţiul disponibil în memorie. Forma **FRE** (" ") forţează calculatorul să facă ordine înainte de a da valoarea spaţiului disponibil.

Observaţie. BASIC-ul nu exploatează decît blocul 0 al memoriei.

Cuvinte cheie asociate. **HIMEM, MEMORY**

GOSUB

GOSUB ⟨număr de linie⟩

```
GOSUB 210
```

Comandă. Apelează un subprogram BASIC cu branşare pe linia indicată. Sfîrşitul subprogramului este marcat prin comanda **RETURN,** trimiţînd programul la instrucţiunea care urmează comenzii **GOSUB.**

Cuvinte cheie asociate. **RETURN**

GOTO

GOTO ⟨număr de linie⟩
GOTO 90

Comandă. Branşează pe numărul de linie indicat.
Cuvinte cheie asociate. Nu există.

GRAPHICS PAPER

GRAPHICS PAPER ⟨cerneală⟩
10 **MODE** 0
20 **MASK** 15
30 **GRAPHICS PAPER** 3
40 **DRAW** 640, 0
run

Comandă. Determină ⟨cerneala⟩ fondului grafic. La trasarea liniilor, fondul nu este vizibil. În exemplul de mai sus, comanda **MASK** permite trasarea unei linii în cratime şi vizualizarea fondului grafic.
Cuvinte cheie asociate. **CLG, GRAPHICS PEN, INK, MASK, TAG, TAGOFF**

GRAPHICS PEN

GRAPHICS PEN [⟨cerneală⟩] [, ⟨tip fond⟩]
10 **MODE** 0
20 **GRAPHICS PEN** 15
30 **MOVE** 200, 0
40 **DRAW** 200, 400
50 **MOVE** 639, 0
60 **FILL** 15
run

Comandă. Stabileşte ⟨cerneala⟩ (între 0 şi 15) pentru trasarea liniilor şi poziţionarea punctelor. Aspectul fondului poate fi, de asemenea, stabilit:
0: Fond opac.
1: Fond transparent.
(Fondul transparent are influenţă asupra fondului grafic al caracterelor scrise cu **TAG** şi asupra spaţiilor cu linii punctate).
Puteţi omite unul din parametri dar nu pe amîndoi. Dacă se omite unul din parametri, valoarea specificată rămîne neschimbată.
Cuvinte cheie asociate. **GRAPHICS PAPER, INK, MASK, TAG, TAGOFF**

HEX$

HEX$ (⟨număr întreg fără semn⟩ [, ⟨întinderea zonei⟩])
PRINT HEX$ (255, 4)
ØØFF

Funcţiune. Transformă un număr întreg într-un număr **HEX**azecimal echivalent, în concordanţă cu numărul de cifre hexazecimale indicat prin ⟨întinderea de zonă⟩ (între 0 şi 16).

⟨Numărul întreg fără semn⟩ care trebuie transformat în formă hexazecimală trebuie să dea o valoare cuprinsă între –32768 şi 65535.

Cuvinte cheie asociate: **BIN$, DEC$, STR$, UNT**

HIMEM

HIMEM
PRINT HIMEM
42619

Funcţiune. Calculează adresa octetului cel mai mare din memorie utilizată de BASIC (care se poate modifica prin comanda **MEMORY**).

Observaţie. BASIC-ul nu exploatează decît blocul **0** din memorie.

Cuvinte cheie asociate. **FRE, MEMORY, SYMBOL, SYMBOL AFTER**

IF

IF ⟨expresie logică⟩ **THEN** ⟨opţiune⟩ [**ELSE** ⟨opţiune⟩]
10 **MODE** 1
20 x=**CINT** (**RND** * 100)
30 **PRINT** "ghiceşte o cifră (de la 0 la 100)"
40 **INPUT** n
50 **IF** n<x **THEN PRINT** n; "este prea mic . . ."
60 **IF** n>x **THEN PRINT** n; "este prea mare . . ."
70 **IF** n=x **THEN** 60 **ELSE** c=c+1 : **GOTO** 40
80 **PRINT** "Bravo!"; "NR. ÎNCERCĂRI="; c+1;

Comandă. Arată dacă ⟨expresia logică⟩ este adevărată; în caz contrar se execută prima ⟨opţiune⟩. Dacă ⟨expresia logică⟩ este falsă, se execută ⟨opţiunea⟩ plasată după **ELSE.** În lipsa celei de-a doua ⟨opţiuni⟩, BASIC-ul trece la linia următoare.

Comenzile **IF THEN** pot fi suprapuse dar se termină la sfîrşitul liniei. Nu putem deci avea declaraţii independente de **IF THEN** pe aceeaşi linie. Atunci cînd rezultatul ⟨expresiei logice⟩ necesită un salt de linie, comanda se poate, de exemplu, formula:

IF a=1 **THEN** 100

. . . sau:

IF a=1 **GOTO** 100

. . . sau:

IF a=1 **THEN GOTO** 100

Cuvinte cheie asociate. **ELSE, GOTO, THEN**

INK

INK ⟨cerneală⟩, ⟨număr de culoare⟩ [, ⟨număr de culoare⟩]
10 **MODE** 1 : **PAPER** 0 : **PEN** 1
20 **FOR** p=0 **TO** 1

```
30 FOR i=0 TO 26
40 INK p, i
50 LOCATE 16, 12 : PRINT "INK"; p; ","; i
60 FOR t=1 TO 400 : NEXT t, i, p
70 INK 0, 1 : INK 1, 24 : CLS
run
```

Comandă. Afectează culoarea sau culorile unei cerneli date. Parametrul ⟨cerneală⟩ furnizează referinţa cernelii (printr-un număr întreg cuprins între 0 şi 15) pentru comenzile **PEN** sau **PAPER** corespunzătoare. Primul parametru ⟨număr de culoare⟩ (întreg) dă o valoare de culoare cuprinsă între 0 şi 26. Dacă cel de-al doilea parametru (facultativ) de culoare este specificat, cerneala trece de la o culoare la alta, cu o viteză definită prin comanda **SPEED INK.**

Cuvinte cheie asociate. **GRAPHICS PAPER, GRAPHICS PEN, PAPER, PEN, SPEED INK**

INKEY

INKEY (⟨număr întreg⟩)

```
10 IF INKEY (55) < > 32 THEN 10
20 PRINT "tocmai aţi tastat [SHIFT] şi V"
30 CLEAR INPUT
```

Funcţiune. Interoghează claviatura pentru indicarea tastelor apăsate. Claviatura este analizată la fiecare cincizeci de secunde. Această funcţiune serveşte la descoperirea poziţiei de sus sau de jos a unei taste prin detectarea valorii −1 (independentă de starea tastelor [**SHIFT**] şi [**CONTROL**]).

În exemplul de mai sus, sistemul detectează acţionarea simultană a lui **SHIFT** şi **V** (număr de tastă 55) înainte de oprirea programului. Numerele de tastă sînt date în diagrama situată în partea din dreapta sus a cutiei calculatorului.

Cuvinte cheie asociate. **CLEAR INPUT, INKEY, JOY**

INKEY$

INKEY$

```
10 CLS
20 PRINT "alege DA sau NU"
30 a$=INKEY$
40 IF a$="" THEN 30
50 IF a$="o" OR a$="O" THEN 80
60 IF a$="n" OR a$="N" THEN 90
70 GOTO 30
80 PRINT "Ai ales DA" : END
90 PRINT "Ai ales NU"
```

Funcţiune. Interoghează claviatura pentru a introduce în program orice şir de caractere. Dacă nu se acţionează nici o tastă a claviaturii, **INKEY$** afişează un şir vid. În exemplul de mai sus, liniile 40 şi 70 dau comanda

unui program de revenire la linia 30 după interogarea claviaturii prin funcţiunea **INKEY$.**

Cuvinte cheie asociate. **CLEAR INPUT, INKEY**

INP

INP (⟨număr de port⟩)
PRINT INP (& FF77)
255

Funcţiune. Citeşte valoarea cuprinsă într-un port de intrări-ieşiri a cărui adresă se transmite prin argumentul acestei funcţii.

Cuvinte cheie asociate. **OUT, WAIT**

INPUT

INPUT [# ⟨număr de canal⟩,] [;] [⟨şir⟩ ⟨separator⟩] ⟨listă de variabile⟩

```
10 MODE 1
20 INPUT "dă-mi două numere de multiplicat (despărţite printr-o virgulă)"; a, b
30 PRINT a; "ori"; b; "fac"; a * b
40 GOTO 20
```

Comandă. Primeşte datele de la canalul precizat (canal #Ø în lipsa specificaţiei).

Un punct-virgulă după **INPUT** anulează trecerea la linie după execuţia comenzii.

⟨Separatorul⟩ poate fi un punct-virgulă sau o virgulă. Un punct-virgulă plasat după şirul de caractere dă naştere unui semn de întrebare; o virgulă îl anulează.

Dacă se efectuează o intrare eronată, de exemplu un Ø pentru un O, BASIC răspunde:

?Redo from start

. . . sau orice alt mesaj de eroare programat de dvs.

Orice răspuns la claviatură trebuie să se termine prin [**RETURN**].

Cuvinte cheie asociate. **LINE INPUT**

INSTR

INSTR ([⟨poziţie de plecare⟩,] ⟨şir de caractere înglobat⟩, ⟨şir de caractere înglobat⟩)

```
10 CLS : FOR n=1 TO 26
20 alfabet$=alfabet$+CHR$ (n+64)
30 NEXT
40 INPUT "introduceţi o literă"; a$
50 b$=UPPER$ (a$)
60 PRINT b$; "este pe poziţie";
70 PRINT INSTR (alfabet$, b$);
80 PRINT "în alfabet." : PRINT
90 GOTO 40
```

Funcţiune. Caută în ⟨şirul de caractere înglobat⟩ apariţia şirului de caractere şi indică poziţia primei apariţii din şirul căutat. În lipsa acestuia, funcţiunea indică valoarea Ø.

Cuvinte cheie asociate. Nu există.

INT

INT (⟨expresie numerică⟩)
PRINT INT (−1.995)
−2

Funcţiune. Rotunjeşte primul număr întreg din partea inferioară, prin îndepărtarea părţii fracţionare. Identic cu **FIX** pentru numerele pozitive, el dă cu 1 mai puţin decît **FIX** pentru numerele negative care nu sînt întregi.

Cuvinte cheie asociate. **CINT, FIX, ROUND**

JOY

JOY (⟨număr întreg⟩)
10 **PRINT** "Pentru a opri programul −";
20 **PRINT** "acţionează maneta de joc"
30 **IF JOY** (Ø)<>Ø **THEN END**
40 **GOTO** 1Ø

Funcţiune. Funcţiunea **JOY** citeşte starea manetei de joc specificată prin ⟨numărul întreg⟩ (Ø sau 1).

Bit	Zecimală
0: Sus	1
1: Jos	2
2: Stînga	4
3: Dreapta	8
4: Tir 2	16
5: Tir 1	32

Cuvinte cheie asociate. **CLEAR, INPUT, INKEY**

KEY

KEY ⟨număr logic de tastă⟩, ⟨şir de caractere alfanumerice⟩
KEY 11, "border 13 : paper Ø : pen 1 : ink Ø, 13 : ink 1, Ø : mode 2 : list"+**CHR$** (13)
Apăsaţi tasta [**ENTER**].

Comandă. Asociază un şir de caractere tastei (**KEY**) care corespunde ⟨numărului logic⟩ de tastă specificată. Există 32 de numere logice de tastă (de la 0 la 31), ocupînd tastele de la 128 la 159. Tastele de la 128 la 140 de pe claviatura numerică se asociază cifrelor de la Ø la 9, punctului zecimal, lui [**RETURN**] şi **RUN** [**RETURN**], dar se pot asocia şi altor şiruri de caractere dacă este necesar.

⟨Numărul logic de tastă⟩ dat prin comanda **KEY** trebuie să fie cuprins între 0 şi 31 sau între 128 şi 159 pentru a corespunde numerelor fizice ale tastelor claviaturii numerice.

Şirul de caractere specificat nu trebuie să depăşească 120 caractere în total. Depăşirea acestei limite antrenează o eroare „Improper argument" (argument incorect).

Cuvinte cheie asociate. **KEY DEF**

KEY DEF

KEY DEF ⟨număr de tastă⟩, ⟨repetiţie⟩ [, ⟨normal⟩ [, ⟨cu şift⟩ [, ⟨cu control⟩]]]
KEY 159, "tasta **TAB**"
KEY DEF 68, 1, 159
Apăsaţi tasta [**TAB**].

Comandă. Defineşte valoarea logică a unei taste (**KEY**) **DEF**inită prin numărul său fizic, cuprins între 0 şi 79.

Cuvinte cheie asociate. **KEY, SPEED KEY**

LEFT$

LEFT$ (⟨şir de caractere alfanumerice⟩, ⟨lungimea cerută⟩)

```
10 CLS
20 a$="AMSTRAD"
30 FOR n=1 TO 7
40     PRINT LEFT$ (a$, n)
50 NEXT
run
```

Funcţiune. Extrage un anumit număr de caractere (între 0 şi 255) la stînga unui ⟨şir de caractere alfanumerice⟩. Dacă şirul de caractere este mai scurt decît se cere, el este utilizat în întregime.

Cuvinte cheie asociate. **MID$, RIGHT$**

LEN

LEN (⟨şir de caractere alfanumerice⟩)

```
10 LINE INPUT "introduceţi o propoziţie"; a$
20 PRINT "propoziţia are o lungime de";
30 PRINT LEN (a$); "caractere."
```

Funcţiune. Dă numărul de caractere din ⟨şirul de caractere alfanumerice⟩, spaţiile cuprinse.

Cuvinte cheie asociate. Nu există

LET

LET ⟨variabilă⟩=⟨expresie⟩
LET x=1ØØ

Comandă. Atribuie o valoare unei variabile. În BASIC-AMSTRAD este suficient să se scrie:

x=1ØØ

Cuvinte cheie asociate. Nu există.

LINE INPUT

LINE INPUT# [⟨număr de canal⟩,] [;] [⟨şir⟩ ⟨separator⟩] ⟨variabilă în şir⟩

10 **LINE INPUT** "tastaţi o linie de text punctată. " a$
20 **CLS**
30 **PRINT** "variabila a este egală cu :—"
40 **PRINT** a$

Comandă. Primeşte o linie întreagă de la canalul indicat (#Ø în lipsa specificaţiei).

⟨Separatorul⟩ poate să fie un punct-virgulă sau o virgulă. Punctul şi virgula antrenează afişarea unui punct de întrebare; virgula îl anulează.

Intrarea lui **LINE INPUT** la claviatură se termină prin activarea tastei [**RETURN**].

LINE INPUT de la canalul #9 al dischetei (sau de la casetă) se termină printr-o întoarcere de car sau prin atribuirea a mai mult de 255 caractere ⟨variabilei şir⟩.

Cuvinte cheie asociate. **INPUT**

LIST

LIST [⟨ansamblu de linii⟩] [, ⟨număr de canal⟩]
LIST 100–1000, #1

Comandă. Listează programul pe canalul dorit. 0 este ecranul, 8 este imprimanta. LISTarea poate să fie întreruptă provizoriu dacă apăsaţi o dată pe tasta [**ESC**], pentru a fi apoi reluată cu ajutorul barei de spaţiu. Dacă apăsaţi de două ori pe [**ESC**], veţi opri listarea şi veţi reveni la modul direct.

Puteţi omite primul sau ultimul număr de linie al parametrului ⟨ansamblu de linii⟩ pentru a lista programul de la început sau pînă la sfîrşit.

Cuvinte cheie asociate. Nu există.

LOAD

LOAD ⟨nume fişier⟩ [, ⟨adresă⟩]
LOAD "fichdisc . xyz", &2AFB

Comandă. Încarcă în memorie un program BASIC de pe dischetă, ştergînd orice alt program. Cu opţiunea ⟨adresa⟩, încarcă un fişier binar la adresa indicată în locul adresei la care se găsea în momentul protejării.

Un program protejat NU poate să fie încărcat prin comanda **LOAD** deoarece el este imediat şters din memorie. În acest caz, utilizaţi **RUN** sau **CHAIN.**

Cuvinte cheie asociate. **CHAIN, CHAIN MERGE, MERGE, RUN, SAVE**

LOCATE

LOCATE [# ⟨număr de canal⟩,] ⟨coordonată x⟩, ⟨coordonată y⟩

```
10 MODE 1
20 FOR n=1 TO 2Ø
30    LOCATE n, n
40    PRINT CHR$ (143); "poziţie";
50    PRINT n; " , "; n
60 NEXT
```

Comandă. Deplasează cursorul de text către o nouă poziţie relativă faţă de colţul superior din partea stîngă a ferestrei (**WINDOW**). #Ø reprezintă canalul prin lipsă.

Cuvinte cheie asociate. **WINDOW**

LOG

LOG (⟨expresie numerică⟩)

```
PRINT LOG (9999)
9.21024037
```

Funcţiune. Calculează **LOG**aritmul natural al unei expresii numerice (mai mare decît 0).

Cuvinte cheie asociate. **EXP, LOG 1Ø**

LOG 1Ø

LOG 1Ø (⟨expresie numerică⟩)

```
PRINT LOG1Ø (9999)
3.99995657
```

Funcţiune. Calculează **LOG**aritmul în bază **1Ø** al ⟨expresiei numerice⟩ (mai mare decît zero).

Cuvinte cheie asociate. **EXP, LOG**

LOWER$

LOWER$ (⟨şir de caractere alfanumerice⟩)

```
10 a$="PRIVIŢI CUM SE SCHIMBĂ LITERELE"
20 PRINT LOWER$ (a$+"DE TIP MINUSCUL")
```

Funcţiune. Schimbă toate majusculele unui şir de caractere alfanumerice în litere mici.

Cuvinte cheie asociate. **UPPER$**

MASK

MASK [⟨număr intreg⟩] [, ⟨trasajul primului punct⟩]

```
10 MODE 0 : INK 5, 21 : INK 8, 16
20 MOVE -100 * RND, 400 * RND
30 WHILE XPOS <640
40 FOR x=1 TO 8
50 MASK 2↑(8-x)
```

60 **DRAWR** 32, 0, x, 1 : **MOVER** –32, 0
70 **NEXT**
80 **MOVER** 34, 0
90 **WEND** : **GOTO** 20

Comandă. Defineşte modelul care trebuie utilizat pentru trasajul liniilor. Valoarea binară a ⟨numărului întreg⟩ cuprinsă între 0 şi 255 activează (1) sau dezactivează (0) bits-ii din fiecare grupă adiacentă de 8 pixeli.

Parametrul ⟨trasajul primului punct⟩ arată dacă primul punct al liniei trebuie să fie trasat (1) sau nu (0).

Puteţi omite unul din parametri dar nu pe amîndoi. Dacă omiteţi unul dintre ei, specificaţia sa rămîne neschimbată.

Cuvinte cheie asociate. **DRAW, DRAWR, GRAPHICS PAPER, GRAPHICS PEN.**

MAX

MAX (⟨listă expresie numerică⟩)
10 n=66
20 **PRINT MAX** (1, n, 3, 6, 4, 3)
run
66

Funcţiune. Calculează valoarea cea mai mare (**MAX**imă) a listei.

Cuvinte cheie asociate. **MIN**

MEMORY

MEMORY ⟨adresă⟩
MEMORY &20AA

Comandă. Defineşte spaţiul de memorie disponibilă fixînd adresa octetului cel mai mare.

Observaţie. BASIC-ul nu exploatează decît blocul 0 al memoriei.

Cuvinte cheie asociate. **FRE, HIMEM, SYMBOL, SYMBOL AFTER**

MERGE

MERGE ⟨nume fişier⟩
MERGE "a a a a a . bas"

Comandă. Încarcă un program de pe dischetă şi-l interclasează cu programul care există deja în memorie.

Numerele de linie ale programului vechi, regăsindu-se în noul program, sînt şterse în mod automat.

Fişierele protejate (protejate prin **SAVE,** p) nu pot fi fuzionate cu un alt program.

Cuvinte cheie asociate. **CHAIN, CHAIN MERGE, LOAD**

MID$

MID$ (⟨şir de caractere alfanumerice⟩, ⟨poziţia de plecare⟩ [,⟨lungimea subşirului⟩])

```
10 MODE 1 : ZONE 3
20 a$="ENCYCLOPEDIE"
30 PRINT "Priviţi cum se silabiseşte"; a$
40 PRINT
50 FOR a=1 TO LEN (a$)
60 PRINT MID$ (a$, n, 1)
70 FOR t=1 TO 700 : NEXT t, 0
80 PRINT : PRINT
90 INPUT "introduceţi un cuvint nou"; a$
100 GOTO 50
```

Funcţiune. Trimite un nou subşir care începe cu ⟨poziţia de plecare⟩ a ⟨şirului de caractere alfanumerice⟩ şi conţine numărul de caractere corespunzător ⟨lungimii subşirului⟩. Dacă parametrul ⟨lungimea subşirului⟩ nu este specificat, funcţiunea afişează restul ⟨şirului de caractere alfanumerice⟩ pornind de la ⟨poziţia de plecare⟩.

Cuvinte cheie asociate. **LEFT$, RIGHT$**

MID$

MID$ (⟨variabilă şir⟩, ⟨poziţie de inserţie⟩ [, ⟨lungimea noului şir⟩])= ⟨nou şir de caractere alfanumerice⟩

```
10 a$="bună ziua"
20 MID$ (a$, 3, 2)="XX"
30 PRINT a$
run
buXX ziua
```

Comandă. Inserează în şirul de caractere specificat un ⟨nou şir de caractere alfanumerice⟩ cu un număr dat de caractere, în ⟨poziţia de inserţie⟩.

Cuvinte cheie asociate. **LEFT$, RIGHT$**

MIN

MIN (⟨listă expresie numerică⟩)

PRINT MIN (3, 6, 2, 999, 8, 9,)

Funcţiune. Dă valoarea cei mai mică (**MIN**imă) din ⟨lista de expresii numerice⟩.

Cuvinte cheie asociate. **MAX**

MOD

⟨argument 1⟩ **MOD** ⟨argument 2⟩

```
PRINT 10 MOD 3
 1
PRINT 10 MOD 5
 0
```

Operator. Calculează restul din diviziunea întreagă a ⟨argumentului 1⟩ prin ⟨argument 2⟩.

Cuvinte cheie asociate. Nu există.

MODE

```
MODE ⟨număr întreg⟩
10 m=m+1 : IF m>2 THEN m=0
20 MODE m
30 PRINT "Modul este acesta"; m.
40 PRINT "Apasă o tastă"
50 IF INKEY$=" " THEN GOTO 50 ELSE 10
```

Comandă. Modifică modul de afişare de pe ecran (0, 1 sau 2) şi restabileşte pe ecran cerneala 0, chiar dacă cerneala folosită în acel moment este diferită. Toate ferestrele şi cursoarele sînt reiniţializate.

Cuvinte cheie asociate. **WINDOW, ORIGIN**

MOVE

```
MOVE ⟨coordonată x⟩, ⟨coordonată y⟩ [, [⟨cerneală⟩] [,⟨tip de cer-
     neală⟩] ]
10 MODE 1 : TAG
20 x=RND * 800-100 : y=RND * 430
30 MOVE x, y
40 PRINT "sînt aici";
50 GOTO 20
```

Comandă. Fixează cursorul grafic la punctul absolut specificat. Parametrul facultativ cerneală (cuprins între 0 şi 15) permite schimbarea culorii stiloului grafic.

Parametrul facultativ ⟨tip de cerneală⟩ determină interacţiunea cernelii pe afişajul care apare pe ecran.

0: Normal
1: **XOR** (SAU exclusiv)
2: **AND** (ŞI)
3: **OR** (SAU)

Cuvinte cheie asociate. **MOVER, ORIGIN, XPOS, YPOS**

MOVER

```
MOVER ⟨deplasare x⟩, ⟨deplasare y⟩ [, [⟨cerneală⟩] [, ⟨tip de cer-
      neală⟩] ]
10 MODE 1 : TAG : MOVE 0, 16
20 PRINT "viaţa are";
30 FOR n=1 TO 10
40 MOVER -45, 16
50 PRINT "suişuri"; : NEXT : PRINT "şi"
60 FOR n=1 TO 10
70 MOVER -64, -16
80 PRINT "coborîşuri" : : NEXT
```

Comandă. Fixează cursorul grafic pe coordonate relative (în raport cu poziţia actuală). Parametrul facultativ ⟨cerneală⟩ (cuprins între 0 şi 15) permite schimbarea culorii stiloului grafic.

Parametrul facultativ ⟨tip de cerneală⟩ determină interacţiunea cernelii pe afişajul apărut pe ecran. Există 4 feluri de cerneală:

0: Normal
1: **XOR** (SAU exclusiv)
2: **AND** (ŞI)
3: **OR** (SAU)

Cuvinte cheie asociate. **MOVE, ORIGIN, XPOS, YPOS**

NEW

NEW
 NEW

Comandă. Şterge programul şi variabilele din memorie.

Cuvinte cheie asociate. Nu există.

NEXT

NEXT [⟨listă de variabile⟩]
10 **FOR** a=1 **TO** 3
20 **FOR** b=0 **TO** 26
30 **MODE** 1
40 **PEN** a : **BORDER** b
50 **PRINT "PEN"**; a; **"BORDER"**; b
60 **FOR** c=1 **TO** 500
70 **NEXT** c, b, a

Comandă. Indică sfîrşitul unei bucle începute cu **FOR.** Comanda **NEXT** poate fi anonimă sau se poate raporta la **FOR**-ul referit. În exemplul de mai sus, ⟨lista de variabile⟩ trebuie să apară în sens invers specificaţiei comenzilor **FOR,** în scopul evitării suprapunerii buclelor imbricate.

Cuvinte cheie asociate. **FOR, STEP, TO**

NOT

NOT ⟨argument⟩
IF NOT "alin"<"bebe" **THEN PRINT** "adevărat" **ELSE PRINT** "fals"
fals
PRINT NOT –1
 0
PRINT NOT 0
 –1

Operator. Execută operaţia SAU asupra numerelor întregi. Inversează fiecare bit din argument.

Cuvinte cheie asociate. **AND, OR, XOR**

ON BREAK CONT

ON BREAK CONT
10 ON BREAK CONT

Comandă. Anulează acţiunea tastei **[ESC]**, împiedicînd oprirea programului. Această comandă trebuie să se execute cu grijă, deoarece programul nu poate fi întrerupt decît prin reiniţializarea completă a calculatorului (trebuie deci să protejaţi programul înainte de a-l lansa).

Puteţi dezactiva **ON BREAK CONT** prin **ON BREAK STOP** din interiorul unui program.

Cuvinte cheie asociate. **ON BREAK GOSUB, ON BREAK STOP**

ON BREAK GOSUB

ON BREAK GOSUB ⟨număr de linie⟩

```
10 ON BREAK GOSUB 40
20 PRINT "programul revine"
30 GOTO 20
40 CLS : PRINT "Apasă de două ori tasta [ESC],";
50 PRINT "apelează subprogramul"
60 FOR t=1 TO 2000 : NEXT
70 RETURN
```

Comandă. Cere BASIC-ului să treacă la subprogramul specificat prin număr de linie atunci cînd apăsaţi de două ori pe **[ESC]**.

Cuvinte cheie asociate. **ON BREAK CONT, ON BREAK STOP, RETURN**

ON BREAK STOP

ON BREAK STOP

```
10 ON BREAK GOSUB 40
20 PRINT "programul revine"
30 GOTO 20
40 CLS : PRINT "Apasă de două ori pe [ESC],";
50 PRINT "apelează subprogramul"
60 FOR t=1 TO 2000 : NEXT
65 ON BREAK STOP
70 RETURN
```

Comandă. Dezactivează comenzile **ON BREAK CONT** şi **ON BREAK GOSUB** pentru a permite oprirea programului la activarea tastei **[ESC]**. În exemplul de mai sus, comanda **ON BREAK GOSUB** nu funcţionează decît o singură dată, deoarece ea se dezactivează la linia 65 în subprogramul **ON BREAK.**

Cuvinte cheie asociate. **ON BREAK CONT, ON BREAK GOSUB**

ON ERROR GOTO

ON ERROR GOTO ⟨număr de linie⟩

```
10 ON ERROR GOTO 60
20 CLS : PRINT "Dacă se găseşte o eroare,";
```

30 **PRINT** "atunci să se **LIST**eze programul"
40 **FOR** t=1 **TO** 4000 : **NEXT**
50 **GOTO** 100
60 **PRINT** "Eroare găsită pe linie";
70 **PRINT ERL** : **PRINT** : **LIST**

Comandă. Se trece la linia specificată în momentul în care este găsită o eroare.

Vezi şi comanda **RESUME.**

Cuvinte cheie asociate. **DERR, ERL, ERR, ERROR, RESUME**

ON ⟨expresie⟩ GOSUB

ON ⟨selector⟩ **GOSUB** ⟨listă cu numere de linie⟩

Comandă. Selecţionează o linie din subprogram în funcţie de valoarea selectorului (număr întreg cuprins între 0 şi 255). Ordinea valorilor selectorilor determină numărul de linie care trebuie extras din lista cu numere de linie. Dacă această expresie este egală cu zero, sau dacă ea este mai mare decît numărul de linii din lista specificată în comandă, selecţia nu mai are loc.

Cuvinte cheie asociate. **RETURN**

ON ⟨expresie⟩ GOTO

ON ⟨selector⟩ **GOTO** ⟨listă cu numere de linie⟩

Comandă. Selecţionează o linie la care programul trebuie să sară în funcţie de valoarea ⟨selectorului⟩(număr întreg cuprins între 0 şi 255). Ordinea valorilor selectorului determină numărul de linie care trebuie extras din ⟨lista cu numere de linie⟩.

Dacă această expresie este egală cu zero sau dacă ea este mai mare decît numărul de linii din lista specificată în comandă, selecţia nu mai are loc.

Cuvinte cheie asociate. Nu există.

ON SQ GOSUB

ON SQ (⟨număr de canal⟩) **GOSUB** ⟨număr de linie⟩

10 **ENV** 1, 15, −1, 1
20 **ON SQ** (1) **GOSUB** 60
30 **MODE** 0 : **ORIGIN** 0, 0, 200, 440, 100, 300
40 **FOR** x=1 **TO** 13 : **FRAME** : **MOVE** 330, 200, x
50 **FILL** x : **NEXT** : **GOTO** 40
60 **READ** s : **IF** s=0 **THEN RESTORE** : **GOTO** 60
70 **SOUND** 1, s, 25, 15, 1
80 **ON SQ** (1) **GOSUB** 60 : **RETURN**
90 **DATA** 50, 60, 90, 100, 35, 200, 24, 500, 0

Comandă. Determină o dezorientare în caz de plasare într-un şir sonor. ⟨Numărul de canal⟩ este un număr întreg care indică una din valorile:

1: pentru canal A
2: pentru canal B
3: pentru canal C

Cuvinte cheie asociate. **RETURN, SOUND, SQ**

OPENIN

OPENIN ⟨nume fişier⟩

Comandă. Deschide un fişier de pe dischetă pentru a citi datele destinate programului din memorie. Fişierul care trebuie deschis trebuie să fie un fişier **ASCII.**

Cuvinte cheie asociate. **CLOSEIN, EOF**

OPENOUT

OPENOUT ⟨nume fişier⟩

Comandă. Deschide pe dischetă un fişier de ieşire.

Cuvinte cheie asociate. **CLOSEOUT**

OR

⟨argument⟩ **OR** ⟨argument⟩

IF "alin" ("bebe" **OR** "ciine") "pisică" **THEN PRINT** "adevărat" **ELSE PRINT** "fals"
adevărat
PRINT 1 **AND** 0
1

Operator. Execută în limbaj maşină operaţii booleene asupra numerelor întregi. Pune 1 în toate cazurile în afară de cel în care cele două argumente sînt egale cu zero.

Cuvinte cheie asociate. **AND, NOT, XOR**

ORIGIN

ORIGIN ⟨x⟩, ⟨y⟩ [, ⟨stînga⟩, ⟨dreapta⟩, ⟨sus⟩, ⟨jos⟩]

10 **MODE** 1 : **BORDER** 13 : **TAG**
20 **ORIGIN** 0, 0, 100, 540, 300, 100
30 **GRAPHICS PAPER** 3 : **CLG**
40 **FOR** x=550 **TO** −340 **STEP** −10
50 **MOVE** x, 206
60 **PRINT** "Iată o fereastră grafică";
70 **FRAME** : **NEXT** : **GOTO** 40

Comandă. Stabileşte punctul de origine al cursorului grafic cu coordonatele ⟨x⟩, ⟨y⟩ specificate. Puteţi, de asemenea, să stabiliţi dimensiunile ferestrei grafice prin specificarea ultimilor patru parametri facultativi. Dacă coordonatele specificate pentru fereastra grafică se află în afara ecranului, marginile ecranului sînt atunci considerate drept limite ale ferestrei.

Cuvinte cheie asociate. **CLG**

OUT

OUT ⟨număr de port⟩, ⟨număr întreg⟩

OUT &F8F4, &FF

Comandă. Trimite valoarea ⟨numărului întreg⟩ (cuprins între 0 şi 255) spre ieşirea precizată prin adresă.

Cuvinte cheie asociate. **INP, WAIT**

PAPER

PAPER [# ⟨număr de canal⟩,] ⟨cerneală⟩

```
10 MODE 0 : PEN 0 : INK 0, 13
20 FOR p=1 TO 15
30 PAPER p : CLS
40 LOCATE 7, 12 : PRINT "PAPER"; p
50 FOR t=1 TO 500 : NEXT t, p
```

Comandă. Stabileşte culoarea fondului ecranului pentru caractere. Din momentul afişajului caracterelor pe ecran, matricea (grila) este completată cu ⟨cerneala⟩ corespunzătoare hîrtiei (**PAPER INK**) înainte ca ea însăşi să fie afişată (în afara modului transparent). Dacă ⟨numărul de canal⟩ nu este specificat, canalul #Ø este luat în lipsă. Numărul de culori disponibile depinde de modul ales.

Cuvinte cheie asociate. **INK, GRAPHICS PAPER, PEN**

PEEK

PEEK (⟨adresă⟩)

```
10 MODE 1 : ZONE 7
20 WINDOW 1, 40, 1, 2 : WINDOW#1, 1, 40, 3, 25
30 PRINT "Adresă memorie"
40 LOCATE 20,1 : PRINT "Conţinut memorie"
50 FOR n=0 TO 65535
60 p=PEEK(n)
70 PRINT#1,n,"(&";HEX$(n);")";
80 PRINT#1,TAB(20);p,"(&";HEX$(p);")"
90 NEXT
```

Funcţiune. Citeşte conţinutul locaţiei de memorie Z80 a cărei ⟨adresă⟩ este indicată între paranteze. Această adresă trebuie să fie cuprinsă între & 0000 şi & FFFF (0 şi 65535). **PEEK** nu operează decît pe memorie vie (**RAM**), niciodată pe memorie moartă (**ROM**) şi furnizează valori cuprinse între & 00 şi & FF (0 şi 255).

Cuvinte cheie asociate. **POKE**

PEN

PEN#[⟨număr de canal⟩,][⟨cerneală⟩][,⟨felul fondului⟩]

```
10 MODE 0: PAPER 0 : INK 0,13
20 FOR p=1 TO 15
```

```
30 PEN p : PRINT SPACE$(47);"PEN";p
40 FOR t=1 TO 500 : NEXT t,p : GOTO 20
```

Comandă. Selecţionează ⟨cerneala⟩ care trebuie utilizată (de la 0 la 15) pentru a scrie pe canalul indicat (în lipsă : #0). Parametrul ⟨felul fondului⟩ poate fi ori transparent (1), ori opac (0). Trebuie să figureze cel puţin unul din ultimii doi parametri. Dacă se omite unul dintre ei, valoarea sa anterioară rămîne neschimbată.

Cuvinte cheie asociate. **PAPER**

PI

PI

```
PRINT PI
3.14159265
```

Funcţiune. Furnizează valoarea raportului circumferinţă/diametru al unui cerc.

Cuvinte cheie asociate. **DEG, RAD**

PLOT

PLOT ⟨coordonată x⟩, ⟨coordonată y⟩[,[⟨cerneală⟩]][,⟨tip de cerneală⟩]]

```
10 MODE 1 : BORDER 0 : PAPER 0 : PEN 1
20 INK 0, 0 : INK 1, 26 : INK 2, 13, 26 : DEG
30 FOR x=1 TO 360 : ORIGIN 320, 200
40 DRAW 50#COS(x), 50#SIN(x), 1
50 PLOT 100# COS(x), 25#SIN(x) : NEXT
60 ORIGIN 0, 0 : t=TIME+700 : WHILE TIME<t
70 PLOT RND#640, RND#400 : WEND
80 PLOT RND#640, RND#400, 2
90 GOTO 90
```

Comandă. Afişează grafic punctul de coordonate x şi y. Se defineşte ⟨cerneala⟩ din acest punct şi cea de pe ecran. Iată cele patru moduri posibile:

0 : Normal
1 : **XOR** (SAU exclusiv)
2 : **AND** (SI) : **OR** (SAU)

Cuvinte cheie asociate. **GRAPHICS PEN, PLOTR**

PLOTR

PLOTR ⟨deplasare x⟩, ⟨deplasare y⟩[,[⟨cerneală⟩]][,⟨tip de cerneală⟩]]

```
10 REM utilizează cursorul pentru a desena
20 BORDER 0 : GRAPHICS PEN 1
30 MODE 1 : PLOT 320, 200
40 IF INKEY(0)=0 THEN PLOTR 0, 1
50 IF INKEY(1)=0 THEN PLOTR 1, 0
60 IF INKEY(2)=0 THEN PLOTR 0, -1
70 IF INKEY(8)=0 THEN PLOTR -1, 0
```

80 **IF INKEY**(9)=0 **THEN** 30 : **REM** [**COPY**]=**CLS**
90 **GOTO** 40

Comandă. Afişează grafic pe ecran punctul de coordonate x şi y relative la poziţia cursorului în acel moment. Se defineşte ⟨cerneala⟩ din acest punct pe o scară cuprinsă între 0 şi 15. Parametrul facultativ tip de cerneală defineşte modul de interacţiune dintre culoarea utilizată şi cea de pe ecran.

Iată cele patru moduri posibile:
0 : Normal
1 : **XOR** (SAU exclusiv)
2 : **AND** (ŞI)
3 : **OR** (SAU)

Cuvinte cheie asociate. **GRAPHICS PEN, PLOT**

POKE

POKE ⟨adresă⟩, ⟨număr întreg⟩
10 **FOR** m=49152 **TO** 65535
20 **POKE** m, 100
30 **NEXT**
run

Comandă. Înscrie valoarea corespunzătoare ⟨numărul întreg⟩ (cuprins între 0 şi 255) direct în memoria vie (**RAM**) din Z80 a cărui ⟨adresă⟩ este indicată.

Comanda se va executa cu atenţie.

Cuvinte cheie asociate. **PEEK**

POS

POS (#⟨număr de canal⟩)

Funcţiune. Calculează poziţia cursorului de text pe axa orizontală, pornind de la marginea stîngă a ferestrei. ⟨Numărul de canal⟩ trebuie obligatoriu să fie precizat; el nu poate lua în schimb valoarea #0.

Cuvinte cheie asociate. **VPOS, WINDOW**

PRINT

PRINT [#⟨număr de canal⟩,] [⟨listă articole de imprimat⟩]

Comandă. Afişează lista de articole de imprimat pe canalul indicat (#0 în lipsă).

Punctul şi virgula arată calculatorului că un articol trebuie să fie imediat imprimat după precedentul.

Virgula arată că un articol trebuie să fie fixat la zona următoare.

PRINT SPC

PRINT TAB
PRINT [# ⟨număr de canal⟩,][⟨listă de articole de imprimat⟩][;]
[**SPC**(⟨număr întreg⟩)][⟨listă de articole de imprimat⟩]

PRINT [#⟨număr de canal⟩,][⟨listă de articole de imprimat⟩][;]
[**TAB** (⟨număr întreg⟩)][⟨listă de articole de imprimat⟩]

```
10 PRINT "aceasta este instrucţiunea SPC"
20 FOR x=4 TO 15
30 PRINT SPC(5)"a"; SPC(x); "b"
40 NEXT
50 PRINT "aceasta este instrucţiunea TAB"
60 FOR x=6 TO 15
70 PRINT TAB(5)"a"; TAB(x); "b"
80 NEXT
```

SPC pregăteşte numărul de spaţii libere indicat prin ⟨număr întreg⟩ înainte de a imprima sau de a afişa articolul indicat, cu condiţia ca acesta din urmă să se menţină pe linie. Este deci inutil să se utilizeze punctul şi virgula cu comanda **SPC**.

TAB pregăteşte, pornind de la marginea stîngă, numărul de spaţii libere indicat înainte de a imprima sau de a afişa articolul desemnat, cu condiţia ca acesta din urmă să fie pe linie. Punctul şi virgula sînt deci inutile după **TAB**. Dacă cursorul a depăşit deja poziţia cerută, înainte de tabulare se efectuează o schimbare de linie.

PRINT USING

PRINT [#⟨număr de canal⟩][⟨listă de articole de imprimat⟩][;]
[**USING** ⟨model de format⟩][⟨separator⟩⟨expresie⟩]

```
10 FOR x=1 TO 10
20 n=100000*(RND↑5)
30 PRINT "marfă"; USING "#########.##"; n
40 NEXT
run
```

PRINT USING permite definirea afişajului unei expresii transmise prin comanda **PRINT**. Pentru aceasta, se defineşte ⟨modelul de format⟩ sub care vrem să apară expresia. Ca ⟨separator⟩ se foloseşte ori o virgulă, ori un punct şi o virgulă. ⟨Modelul de format⟩ este un şir de caractere alcătuit din "indicatori de cîmp", şi anume:

Formate numerice

Într-un număr:

\# Fiecare semn # arată amplasarea unei cifre.
Exemplu: ######

. Indică amplasarea punctului zecimal (echivalent cu virgula noastră).
Exemplu: ###### .##

, (Rezervă un spaţiu). Acest semn, care nu poate să figureze decît imediat înaintea punctului zecimal, arată că cifrele situate la stînga punctului zecimal vor fi dispuse în grupe de trei separate între ele printr-o virgulă.
Exemplu: ######,.##

Încadrarea unui număr:

££ (Rezervă două spaţii). Arată că semnul £ va apare imediat înaintea primei cifre sau punct zecimal, adică pe una din poziţiile rezervate cifrelor.
Exemplu: ££######,.##

** (Rezervă două spaţii). Arată că toate spaţiile libere situate înaintea numărului se vor completa cu asteriscuri.
Exemplu: **######,.##

(Rezervă trei spaţii). Adaugă opţiunile ** şi ££, adică asteriscurile în faţă şi semnul £ care precede imediat numărul.

$$ (Rezervă două spaţii). Arată că semnul $ va apare imediat la stînga primei cifre sau a punctului zecimal, adică pe una din poziţiile rezervate cifrelor.
Exemplu: $$######,.##

**$ (Rezervă trei spaţii). Adaugă opţiunile ** şi $$, adică asteriscurile în faţă şi semnul $ care precede imediat numărul.
Exemplu: **$######,.##

+ Arată că dorim să vedem apărînd semnul numărului. Acest semn va apare înaintea numărului dacă + este situat la începutul modelului de format şi după număr dacă este situat la sfîrşit.
Exemplu: +######.####

– Semnul – nu poate figura decît la sfîrşitul expresiei.
Prezenţa sa este necesară după orice număr sau orice exponent negativ. În lipsa acestei specificaţii, semnul – apare înaintea numărului negativ.
Exemplu: ####.####–

↑↑↑↑ Arată că numărul trebuie să apară în exponent. Semnele se aşează după ultima poziţie a cifrelor, dar înainte de orice semn + sau – final.
Exemplu: #.####↑↑↑↑+

Lungimea maximă a ⟨modelului de format⟩ al unui număr este de 20 caractere. Numerele sînt rotunjite la numărul de semne indicat.
Dacă un format este prea mic pentru a conţine expresia:

PRINT USING "####"; 12345678

... aceasta nu se trunchiază, ci apare în întregime, precedată de semnul % care indică un "format eronat".

Formatul unui şir de caractere alfanumerice

```
10 CLS : a$="abcdefghijklmnoprst"
20 PRINT "şir de caractere alfanum.="; a$
30 PRINT : PRINT "Cu ! ="; a$
40 PRINT USING "!"; a$
50 PRINT : PRINT "Cu\spaţii\=";
60 PRINT USING "\        \"; a$
70 PRINT : PRINT "Cu&=";
80 PRINT USING "&"; a$
90 GOTO 90
run
```

! Arată că numai primul caracter al şirului trebuie să apară.
Exemplu: !

\<spaţii>\
Arată că numai primele caractere x ale şirului trebuie să apară, x fiind egal cu lungimea formatului (inclusiv barele).
Exemplu: \ \

& Arată că şirul trebuie să apară "aşa cum este".
Exemplu: &

<Modelul de format> poate să fie reprezentat printr-o variabilă alfanumerică, aşa cum ne arată exemplul care urmează:

```
10 a$="FF######,.##"
20 b$="!"
30 PRINT USING a$; 12345.6789;
40 PRINT USING b$; "centime"
run
```

Cuvinte cheie asociate. **SPC, TAB, ZONE**

RAD

RAD
RAD

Comandă. Stabileşte modul de calcul în **RAD**iani. În BASIC, acesta este adoptat prin lipsă.

Cuvinte cheie asociate. **ATN, COS, DEG, SIN, TAN**

RANDOMIZE

RANDOMIZE [<expresie numerică>]

RANDOMIZE 123.456
PRINT RND
0.258352139

Comandă. Dă o valoare aleatoare calculată pornind de la "expresia numerică" indicată. Generatorul de numere aleatoare furnizează o secvenţă pseudoaleatorie în care fiecare număr depinde de precedentul. Secvenţa însăşi este predeterminată. Dacă valoarea iniţială nu este precizată în comandă, utilizatorul este cel care o va introduce în execuţie. **RANDOMIZE TIME** furnizează o secvenţă practic imprevizibilă.

Cuvinte cheie asociate. **RND**

READ

READ <listă de variabile>

Comandă. Citeşte datele conţinute într-o instrucţiune **DATA** atribuindu-le unor variabile. **READ** trece automat de la o dată la următoarea. Comanda **RESTORE** permite să se revină la o instrucţiune **DATA** anterioară.

Cuvinte cheie asociate. **DATA, RESTORE**

RELEASE

RELEASE ⟨canale sonore⟩
10 **SOUND** 65, 1000, 100
20 **PRINT** "apasă [R] pentru a elibera nota"
30 **IF INKEY**(50)=−1 **THEN** 30
40 **RELEASE** 1

Comandă. Eliberează canalele sonore blocate prin comanda **SOUND.**

Parametrul ⟨canale sonore⟩ ia următoarele valori:

1 : Eliberează canalul A
2 : Eliberează canalul B
3 : Eliberează canalele A şi B
4 : Eliberează canalul C
5 : Eliberează canalele A şi C
6 : Eliberează canalele B şi C
7 : Eliberează canalele A, B şi C

Cuvinte cheie asociate. **SOUND**

REM

REM text
10 **REM** CHASSE AUX ENVAHISSEURS DANS L'HYPERSPACE INTER-GALACTIQUE
20 **REM** **COPYRIGHT** by **AMSOFT**

Comandă. Inserează comentarii în program. BASIC nu ţine seama de ⟨textul⟩ situat pe linia din dreapta lui **REM**, chiar dacă acesta conţine un separator de instrucţiuni " : " sau orice alt cod. : **REM** poate fi înlocuit printr-un apostrof (în toate cazurile), în afară de interiorul unei instrucţiuni **DATA.**

Cuvinte cheie asociate. Nu există.

REMAIN

REMAIN (⟨număr de cronometru⟩)
70 **PRINT** "rămîn"; **REMAIN**(1); "unităţi de timp la crono 1"
run

Funcţiune. Citeşte timpul care rămîne de măsurat cu cronometrul indicat (de la 0 la 3), înainte de a-l dezactiva.

Cuvinte cheie asociate. **AFTER, DI, EI, EVERY**

RENUM

RENUM [⟨număr de linie nou⟩],[⟨număr de linie vechi⟩][,⟨increment⟩]]
10 **CLS**
20 **REM** această linie va deveni: linia 123
30 **REM** această linie va deveni: linia 124

40 **REM** această linie va deveni: linia 125
RENUM 123, 20, 1
LIST

Comandă. **RENUM**erotează liniile unui program.

⟨Numărul de linie vechi⟩ este un parametru care indică linia programului cu care dorim să incepem numerotarea. În lipsa acestui parametru, toate liniile programului se vor renumerota.

⟨Numărul de linie nou⟩ indică noul număr al primei linii renumerotate (prin lipsă 10). ⟨Incrementul⟩ indică spaţiul dorit între două linii (prin lipsă 10). **RENUM** operează reajustările necesare în interiorul instrucţiunilor de apel precum **GOTO** şi **GOSUB**. În schimb, lasă neschimbate numerele de linii cuprinse în şiruri de caractere care apar în comenzile **KEY, REM, CHAIN, CHAIN MERGE.** Numerele de linie trebuie să fie cuprinse între 1 şi 65535.

Cuvinte cheie asociate. **DELETE, LIST**

RESTORE

RESTORE [⟨număr de linie⟩]

Comandă. Repune indicatorul pe instrucţiunea **DATA** indicată. În lipsa parametrului, indicatorul revine la prima instrucţiune **DATA** din program.

Cuvinte cheie asociate. **DATA, READ**

RESUME

RESUME [⟨număr de linie⟩]

Comandă. Reia execuţia unui program după descoperirea şi tratarea unei erori prin comanda **ON ERROR GOTO.** Dacă nu se indică nici un ⟨număr de linie⟩, execuţia programului se reia la linia care conţine eroarea descoperită.

Cuvinte cheie asociate. **DERL, ERL, ERR, ERROR, ON ERROR GOTO, RESUME NEXT**

RESUME NEXT

RESUME NEXT

Comandă. Reia execuţia unui program după detectarea şi tratarea unei erori prin comanda **ON ERROR GOTO.**

Execuţia programului se reia începînd de la linia care urmează imediat liniei eronate.

Cuvinte cheie asociate. **DERR, ERR, ERROR, ON ERROR GOTO, RESUME**

RETURN

RETURN

10 **GOSUB** 50 : **PRINT** "După **GOSUB**" : **END**
50 **FOR** n=1 **TO** 20

```
60 PRINT "subprogram"
70 NEXT : PRINT
80 RETURN
run
```

Comandă. Indică sfîrşitul unui subprogram. După execuţia unui subprogram, se revine la instrucţiunea care urmează imediat după **GOSUB**-ul corespunzător.

Cuvinte cheie asociate. **GOSUB**

RIGHT$

RIGHT$ (⟨şir de caractere alfanumerice⟩, ⟨lungimea cerută⟩)

```
10 MODE 1 : a$="calculator CPC 6128"
20 FOR n=1 TO 16 : LOCATE 41-n, n
30 PRINT RIGHT$ (a$, n)
40 NEXT
run
```

Funcţiune. Extrage un anumit număr de caractere (între 0 şi 255) de la stînga unui ⟨şir de caractere alfanumerice⟩. Dacă şirul este mai scurt decît ⟨lungimea cerută⟩, el este utilizat în întregime.

Cuvinte cheie asociate. **LEFT$, MID$**

RND

RND [(⟨expresie numerică⟩)]

```
10 RANDOMIZE
20 FOR x=1 TO -1 STEP -1
30 PRINT "parametrul RND="; x
40 FOR n=1 TO 6
50 PRINT RND (x)
60 NEXT n, x
run
```

Funcţiune. Furnizează numărul următor al secvenţei pseudoaleatorii în curs atunci cînd ⟨expresia numerică⟩ este pozitivă sau cînd ea nu figurează în comandă.

Atunci cînd ⟨expresia numerică⟩ este nulă, **RND** trimite ultimul număr generat.

O valoare negativă a ⟨expresiei numerice⟩ lansează o nouă secvenţă aleatorie, al cărei **RND** dă primul element.

Cuvinte cheie asociate. **RANDOMIZE**

ROUND

ROUND (⟨expresie numerică⟩ [,⟨număr de zecimale⟩])

```
10 FOR n=4 TO -4 STEP -1
20 PRINT ROUND (1234·5678, n),
30 PRINT "rotunjit la"; n; "zecimale"
40 NEXT
run
```

Funcţiune. Rotunjeşte ⟨expresia numerică⟩ la numărul de cifre după virgulă sau la numărul de puteri indicat prin parametrul ⟨număr de zecimale⟩. Dacă acest parametru este negativ, expresia se rotunjeşte la un număr întreg absolut, urmat de un număr de zerouri egal cu valoarea sa absolută.

Cuvinte cheie asociate. **ABS, CINT, FIX, INT**

RUN

RUN ⟨şir de caractere alfanumerice⟩
RUN "disc"

Comandă. Încarcă şi execută un program BASIC sau un program-obiect de pe dischetă. Orice program prezent deja în memorie este şters automat.

Această comandă permite accesul direct la programele BASIC protejate.

Cuvinte cheie asociate. **LOAD**

RUN

RUN [⟨număr de linie⟩]
RUN 2ØØ

Comandă. Execută programul BASIC prezent în memorie, începînd cu ⟨numărul de linie⟩ indicat sau, în lipsa acestuia, cu începutul programului. **RUN** reiniţializează toate variabilele.

Această comandă poate să nu permită accesul la programele protejate încărcate în memorie.

Cuvinte cheie asociate. **CONT, END, STOP**

SAVE

SAVE ⟨nume fişier⟩ [,⟨tip de fişier⟩] [,⟨parametri binari⟩]
SAVE "fichdisc · xyz"
. . . salvează fişierul în mod BASIC neprotejat
SAVE "fichdisc · zyz", p
. . . salvează fişierul în mod BASIC protejat.
SAVE "fichdisc · xyz", A
. . . salvează fişierul în mod ASCII.
SAVE "fishdisc · xyz", B, 8000, 3000, 8001
. . . salvează fişierul în mod Binar. În exemplul nostru, programul va fi înmagazinat în memorie pornind de la adresa 8000 şi va ocupa 3000 octeţi. Adresa (facultativă) din punctul de intrare este 8001.

Comandă. Salvează pe dischetă programul care se găseşte momentan în memorie. O zonă din memorie încărcată pe dischetă se numeşte fişier Binar. Iată care sînt diferiţii parametri binari:

⟨adresa de început⟩, ⟨dimensiunea fişierului⟩ [,⟨punct de intrare⟩]

Este posibil să salvăm memoria ecran sub formă de fişier binar. Această operaţie, denumită "vidaj de ecran" se realizează cu ajutorul comenzii următoare:

SAVE "ecran", B, &C000, &4000

Se obţine apoi reîncărcarea sa pe ecran prin

LOAD "ecran"

Cuvinte cheie asociate. **CHAIN, CHAIN MERGE, LOAD, MERGE, RUN**

SGN

SGN (⟨expresie numerică⟩)

Funcţiune. Stabileşte semnul ⟨expresiei numerice⟩. **SGN** trimite înapoi valorile: −1 (dacă expresia este negativă), 0 (dacă ea este nulă) şi 1 (dacă ea este pozitivă).

Cuvinte cheie asociate. **ABS**

SIN

SIN (⟨expresie numerică⟩)

Funcţiune. Calculează **SIN**usul ⟨expresiei numerice⟩ indicate. Argumentul se poate exprima în grade sau în radiani utilizînd, respectiv, funcţiunile **DEG** şi **RAD.**

Cuvinte cheie asociate. **ATN, COS, DEG, RAD, TAN**

SOUND

SOUND ⟨stare de canal⟩, ⟨perioadă sonoră⟩ [,⟨durată⟩ [,⟨volum⟩ [,⟨anvelopă de volum⟩ [,⟨anvelopă de tonalitate⟩ [,⟨perioada zgomotului⟩]]]]]

```
10 FOR z=0 TO 4095
20    SOUND 1, z, 1, 12
30 NEXT
```

Comandă. Autorizează programarea unui sunet, cu ajutorul următorilor parametri:

Parametrul 1: stare de canal

⟨Stare de canal⟩ admite drept valoare numere întregi cuprinse între 1 şi 255.

Cuvinte cheie asociate. **ENT, ENV, ON SQ GOSUB, RELEASE, SQ**

SPACE$

SPACES (⟨număr întreg⟩)

Funcţiune. Creează un şir de spaţii cu lungimea indicată (de la 0 la 255).

Cuvinte cheie asociate. **SPC, STRING$, TAB**

SPC

(Vezi **PRINT SPC**)

SPEED INK

SPEED INK ⟨perioada 1⟩, ⟨perioada 2⟩

```
10 BORDER 7, 18
20 FOR i=30 TO 1 STEP -1
30 SPEED INK i, i
40 FOR t=1 TO 700 : NEXT t, i
```

Comandă. Permite stabilirea perioadei de alternanță atunci cînd o instrucțiune **INK** sau **BORDER** prescrie utilizarea a două culori intermitente. Duratele respective de utilizare ale primei și ale celei de-a doua culori sînt indicate în 50 zecimi de secunde prin parametrii ⟨perioada 1⟩ și ⟨perioada 2⟩.

În momentul alegerii parametrilor, gîndiți-vă la riscurile unor efecte secundare hipnotice!

Cuvinte cheie asociate. **BORDER, INK**

SPEED KEY

SPEED KEY ⟨interval inițial⟩, ⟨interval dintre repetiții⟩

```
10 CLS: FOR k=7 TO 1 STEP -2
20 PRINT "Dă-ți numele, apoi RETURN"
30 SPEED KEY k, k
40 LINE INPUT a$ : NEXT
50 PRINT "ce nume caraghios!"
```

Comandă. Stabilește viteza de repetiție automată a claviaturii. Parametrul ⟨interval inițial⟩ stabilește timpul de reacție (măsurat în 50 de zecim de secundă) dintre apăsarea pe tastă și inceputul repetiției automate. ⟨Intervalul dintre repetiții⟩ stabilește timpul scurs care separă repetițiile.

Cuvinte cheie asociate. **KEY DEF**

SPEED WRITE

SPEED WRITE ⟨număr intreg⟩

```
SPEED WRITE 1
```

Comandă. Arată viteza de transmitere a datelor calculatorului către un cititor de casete. Această viteză este fie de 2000 bits pe secundă dacă parametrul este egal cu 1, fie, prin lipsă, de 1000 bits pe secundă dacă acesta este egal cu 0. În timpul încărcării unui fișier înregistrat pe casetă, calculatorul alege în mod automat viteza potrivită de citire.

SPEED WRITE Ø este debitul care asigură cea mai bună fiabilitate de transfer.

Comanda **SPEED WRITE** nu se aplică unităților de dischete.

Cuvinte cheie asociate. **OPENOUT, SAVE**

SQ

SQ (⟨număr de canal⟩)

```
10 SOUND 65, 100, 100
20 PRINT SQ (1)
run
67
```

Funcţiune. Indică starea firului de aşteptare dintr-un canal sonor dat. ⟨Numărul de canal⟩ trebuie să fie o expresie numerică întreagă care ia următoarele valori:

1: canal A; 2: canal B; 4: canal C

Cuvinte cheie asociate. **ON SQ GOSUB, SOUND**

SQR

SQR (⟨expresie numerică⟩)

```
PRINT SQR (9)
3
```

Funcţiune. Furnizează rădăcina pătrată a expresiei numerice indicate.

Cuvinte cheie asociate. Nu există.

STEP

(Vezi **FOR**)

STOP

STOP

```
10 FOR n=1 TO 30 : PRINT n : NEXT
20 STOP
30 FOR n=31 TO 60 : PRINT n : NEXT
run
cont
```

Comandă. Întrerupe un program, lăsînd utilizatorului posibilitatea de a relua execuţia acestuia cu ajutorul comenzii **CONT. STOP;** permite astfel întreruperea unui program într-un punct dat în scopul realizării unei puneri la punct.

Cuvinte cheie asociate. **CONT, END**

STR$

STR$ (⟨expresie numerică⟩)

Funcţiune. Furnizează sub formă de şir de caractere alfanumerice reprezentarea zecimală a ⟨expresiei numerice⟩ indicate.

Cuvinte cheie asociate. **BIN$, DEC$, HEX$, VAL**

STRING$

STRING$ (⟨lungime⟩, ⟨caracter⟩)
PRINT STRING$ (40, "*")

Funcţiune. Furnizează un şir de caractere cu lungimea indicată (între 0 şi 255), alcătuit prin repetiţia aceluiaşi caracter.

Cuvinte cheie asociate. **SPACE$**

SWAP

(Vezi **WINDOW SWAP**)

SYMBOL

SYMBOL ⟨număr de caracter⟩, ⟨listă linie⟩

```
10 MODE 1 : SYMBOL AFTER 105
20 rînd1=255 : REM 1111111 în binar
30 rînd2=129 : REM 10000001 în binar
40 rînd3=189 : REM 10111101 în binar
50 rînd4=153 : REM 10011001 în binar
60 rînd5=153 : REM 10011001 în binar
70 rînd6=189 : REM 10111101 în binar
80 rînd7=129 : REM 10000001 în binar
90 rînd8=255 : REM 1111111 în binar
100 PRINT "Linia 110 redefineşte litera i (105)"
110 SYMBOL 105, rînd1, rînd2, rînd3, rînd4, rînd5, rînd6, rînd7,
    rînd8
```

Comandă. Redefineşte forma unui caracter afişat pe ecran. Fiecare parametru ia o valoare întreagă situată între 0 şi 255.

Înainte de a fi în măsură să atribuie un loc în memorie unui caracter redefinit, calculatorul trebuie să aibă pregătită comanda:

SYMBOL AFTER x

. . . unde x este mai mic sau egal cu numărul de caracter care trebuie redefinit.

Se introduce apoi comanda **SYMBOL**, urmată imediat de numărul de caractere x.

După **SYMBOL** x apar cei opt parametri care definesc unul cîte unul cele opt linii care alcătuiesc caracterul, începînd cu cel de sus. Fiecare dintre parametri ia o valoare cuprinsă între 0 şi 255.

Cuvinte cheie asociate. **HINEM, MEMORY, SYMBOL AFTER**

SYMBOL AFTER

SYMBOL AFTER ⟨număr întreg⟩

```
10 CLS
20 SYMBOL AFTER 115
30 PRINT "Linia 40 redefineşte litera s";
40 SYMBOL 115, 0, 56, 64, 64, 48, 8, 8, 112
```

```
50 PRINT "în s"
60 PRINT "se revine la starea normală tastînd:"
70 PRINT "SYMBOL AFTER 240"
run
```

Comandă. Stabileşte limita inferioară a numerelor de caractere redefinibile (de la 0 la 255). Valoarea prin lipsă a numărului întreg este de 240, caz în care se dispune de 16 caractere redefinibile (între 240 şi 255). Atunci cînd ⟨numărul întreg⟩ are ca valoare 32, toate caracterele aflate între 32 şi 255 sînt redefinibile. Comanda SYMBOL AFTER 256 interzice deci orice redefinire de caracter.

Comanda **SYMBOL AFTER** restabileşte valoarea prin lipsă a tuturor caracterelor redefinite mai înainte.

Comanda **SYMBOL AFTER** nu poate să funcţioneze dacă valoarea lui **HIMEM** a fost modificată anterior printr-o comandă **MEMORY** sau prin deschiderea unui fişier cu ajutorul lui **OPENIN** sau **OPENOUT.** Calculatorul afişează în acest caz mesajul de eroare "Improper argument" (numai dacă starea precedentă nu este **SYMBOL AFTER** 256).

Cuvinte cheie asociate. **HIMEM, MEMORY, SYMBOL**

TAB

(Vezi **PRINT TAB**).

TAG

TAG [#⟨număr de canal⟩]

```
10 INPUT "introdu-ţi numele"; a$ : CLS
20 PRINT "Ce du-te vino"; a$; " !! "
30 TAG
40 x=LEN (a$) #17 : y=50+RND # 300 : MOVE -x, y
50 FOR f=-x TO 640 STEP RND #7+3
60 MOVE b, y : PRINT a$; " "; : FRAME : NEXT
70 FOR b=640 TO -x STEP-RND#7+3
80 MOVE b, y : PRINT a$; " " ; : FRAME : NEXT
90 GOTO 40
```

Comandă. Scrie textul specificat la poziţia cursorului grafic. Această comandă permite introducerea textului şi a simbolurilor pixel cu pixel (mai degrabă decît caracter cu caracter). Numărul de canal ia prin lipsă valoarea #Ø.

Extremitatea stîngă a şirului de caractere se situează pe cursorul grafic. Caracterele de control nevizualizate, precum schimbarea de linie sau întoarcerea de car nu vor avea nici un efect pe ecran dacă instrucţiunea **PRINT** se termină prin punct şi virgulă; în caz contrar, vor apare sub forma lor grafică.

Dacă indicatorul de canal este #Ø (prin lipsă), BASIC-ul anulează comanda **TAG** în momentul întoarcerii în mod direct.

Cuvinte cheie asociate. **TAGOFF**

TAGOFF

TAGOFF [#⟨număr de canal⟩]

```
10 MODE 2 : TAG : REM text cu coordonate grafice
20 anul=1984 : FOR x=1 TO 640 STEP 60
30 MOVE x, 400 : DRAWR 0, -350
40 anul=anul+1 : PRINT anul; : NEXT
50 TAGOFF : REM revenire la coordonatele text
60 LOCATE 28, 25 : PRINT "cifre anuale"
70 GOTO 70
```

Comandă. Anulează comanda **TAG** care cuprinde canalul indicat (#Ø prin lipsă). Textul se află deci din nou îndreptat spre poziţia cursorului din text.

Cuvinte cheie asociate. **TAG**

TAN

TAN (⟨expresie numerică⟩)

```
PRINT TAN (45)
1.61977519
```

Funcţiune. Calculează **TAN** genta ⟨expresiei numerice⟩, care trebuie să fie cuprinsă între −200000 şi +200000.

Argumentul se poate exprima în grade sau radiani prin intermediul funcţiunilor **DEG** şi **RAD**.

Cuvinte cheie asociate. **ATN, COS, DEG, RAD, SIN**

TEST

TEST (⟨coordonată x⟩, ⟨coordonată y⟩)

```
10 CLS
20 PRINT "Utilizează stiloul nr : " ;
30 PRINT TEST (12, 394)
40 PRINT "Schimbă modul şi stiloul";
50 PRINT " . . . apoi RUN"
```

Funcţiune. Fixează cursorul grafic la poziţia definită prin x şi y şi indică valoarea parametrului cerneală în acest loc.

Cuvinte cheie asociate. **MOVE, MOVER, TESTR, XPOS, YPOS**

TESTR

TESTR (⟨deplasare x⟩, ⟨deplasare y⟩)

```
10 MODE 0 : FOR x=1 TO 15 : LOCATE 1, x
20 PEN x : PRINT STRING$ (10, 143); : NEXT
30 MOVE 200, 400 : PEN 1
40 FOR n=1 TO 23 : LOCATE 12, n
50 PRINT "stilou"; TESTR (0, -16) : NEXT
```

Funcţiune. Fixează cursorul într-o poziţie de coordonate x şi y în raport cu poziţia sa actuală şi indică valoarea parametrului ⟨cerneală⟩ în acest punct.

Cuvinte cheie asociate. **MOVE, MOVER, TEST, XPOS, YPOS**

THEN

(Vezi **IF**)

TIME

TIME

```
10 CLS : REM ceas
20 INPUT "oră"; oră
30 INPUT "minut"; minut
40 INPUT "secundă"; secundă
50 CLS : dată=INT(TIME/300)
60 WHILE oră<13
70 WHILE minut<60
80 WHILE tic<60
90 tic=(INT (TIME/300)-dată)+secundă
100 LOCATE 1, 1
110 PRINT USING "##"; oră; minut; tic
120 WEND
130 tic=0 : secundă=0 : minut=minut+1
140 GOTO 50
150 WEND
160 minut=0 : oră=oră+1
170 WEND
180 oră=1
190 GOTO 60
```

Funcţiune. Indică timpul scurs de la punerea sub tensiune a calculatorului sau de la ultima comandă **RESET.**

Fiecărei secunde îi corespunde o dată valoarea: **TIME/300.**

Cuvinte cheie asociate. **AFTER, EVERY, WEND, WHILE**

TO

(Vezi **FOR**)

TROFF
TRON

TROFF
TRON

```
10 TROFF : PRINT : PRINT "TROFF"
20 FOR n=1 TO 8
30 PRINT "Programul se întoarce" : NEXT
```

```
40 IF f=1 THEN END
50 TRON : PRINT : PRINT "TRON"
60 f=1 : GOTO 20
```

Comandă. Permite urmărirea execuției unui program prin afișarea fiecărui număr de linie executată. Acest număr este afișat între croșete [ ]. Această funcțiune se obține cu ajutorul comenzii **TRON.** Comanda **TROFF** restabilește modul normal de execuție. Comanda **TRON** este deosebit de valoroasă atunci cînd dorim să urmărim linie cu linie derularea unui program în scopul corectării unei erori.

Cuvinte cheie asociate. Nu există.

UNT

UNT (⟨adresă⟩)

```
PRINT UNT (&FF66)
-154
```

Comandă. Transformă argumentul într-un număr întreg cu semn cuprins −32768 și 32767.

Cuvinte cheie asociate. **CINT, FIX, INT, ROUND**

UPPER$

UPPER$ (⟨șir de caractere alfanumerice⟩)

```
10 CLS : a$="micuții mei, ce-ați crescut!"
20 PRINT UPPER$ (a$)
run
```

Funcțiune. Recopiază ⟨șirul de caractere alfanumerice⟩ indicat, înlocuind cu majuscule caracterele alfabetice (de la A la Z) care apar cu litere mici. Această funcțiune se folosește îndeosebi pentru tratarea intrărilor în care se găsesc amestecate majuscule și litere mici.

Cuvinte cheie asociate. **LOWER$**

USING

(Vezi **PRINT USING**)

VAL

VAL (⟨șir de caractere⟩)

```
10 CLS : PRINT "Îmi cunosc puterile!"
20 PRINT : PRINT "apasă o tastă (1-9)"
30 a$=INKEY$ : IF a$=" " THEN 30
40 n=val (a$) : IF n<1 OR n>9 THEN 30
50 FOR x=1 TO 12
60 PRINT n; "X"; x; "="; n#x
70 NEXT : GOTO 20
```

Funcţiune. Transmite **VAL**oarea numerică a primului sau a primelor caractere (inclusiv semnul negativ şi punctul zecimal) ale ⟨şirului de caractere alfanumerice⟩ indicat.

Obţinem valoarea 0 atunci cînd primul caracter al şirului nu este o cifră. Dacă semnul « – » apare ca prim caracter sau dacă acesta este un punct zecimal urmat de un caracter nenumeric, mesajul de eroare «Type mismatch» (eroare de bătaie) va apare afişat pe ecran.

Cuvinte cheie asociate. **STR$**

VPOS

VPOS #(⟨număr de canal⟩)

```
 10 MODE 1 : BORDER 0 : LOCATE 8, 2
 20 PRINT "utilizează tastele săgeţi (sus/jos)"
 30 WINDOW 39, 39, 1, 25 : CURSOR 1, 1
 40 LOCATE 1, 13
 50 IF INKEY (0)<>-1 THEN PRINT CHR$ (11);
 60 IN INKEY (2)<>-1 THEN PRINT CHR$ (10);
 70 LOCATE #1, 3, 24
 80 PRINT #1,"cursor text";
 90 PRINT #1, "poziţie verticală=";
100 PRINT #1, VPOS (#0) : GOTO 50
```

Funcţiune. Indică, pe axa verticală, poziţia cursorului de text, pornind de la marginea superioară a ferestrei din text. Indicatorul de canal trebuie să apară obligatoriu; el nu ia prin lipsă valoarea #Ø.

Cuvinte cheie asociate. **POS, WINDOW**

WAIT

WAIT ⟨număr de port⟩, ⟨mască⟩, [,⟨inversiune⟩]

```
WAIT FF34, 20, 25
```

Comandă. Determină o aşteptare pînă portul de intrări-ieşiri desemnat transmite o valoare cuprinsă între 0 şi 255, în aşa fel încît după ce a operat un **XOR** (SAU exclusiv) cu ⟨masca⟩, apoi un **AND** (ŞI) cu parametrul de ⟨inversiune⟩, obţinem un rezultat nenul.

BASIC-ul aşteaptă pînă cînd condiţia este verificată.

Această comandă trebuie utilizată cu atenţie.

Cuvinte cheie asociate. **INP, OUT**

WEND

WEND

```
WEND
```

Comandă. Indică sfîrşitul unei secţiuni din program executate în interiorul unei bucle **WHILE.** BASIC recunoaşte automat comanda **WHILE** căreia i se asociază **WEND.**

Cuvinte cheie asociate. **TIME, WHILE**

WHILE

WHILE ⟨expresie logică⟩

10 **CLS** : **PRINT** "cronometru de 10 secunde"; t=**TIME**
20 **WHILE TIME** <t+3000
30 **SOUND** 1, 0, 100, 15
40 **WEND** : **SOUND** 129, 40, 30, 15

Comandă. Repetă o secţiune din program atîta timp cît o condiţie dată este verificată. Cuvîntul cheie **WHILE** indică începutul secţiunii de executat în timp ce ⟨expresia logică⟩ defineşte condiţia de verificat.

Cuvinte cheie asociate. **TIME, WEND**

WIDTH

WIDTH ⟨număr întreg⟩

WIDTH 40

Comandă. Indică numărul de caractere de pe linie în momentul unei ieşiri pe imprimantă.

În absenţa unei comenzi **WIDTH,** calculatorul adoptă prin lipsă valoarea 132.

Cuvinte cheie asociate. **POS**

WINDOW

WINDOW [#⟨număr de canal⟩,] ⟨stînga⟩, ⟨dreapta⟩, ⟨sus⟩, ⟨jos⟩

```
10 MODE 0 : BORDER 0 : REM obiectiv tv
20 INK 0, 0 : INK 1, 25 : INK 2, 23 : INK 3, 21
30 INK 4, 17 : INK 5, 6 : INK 6, 2 : INK 7, 26
40 PAPER 0 : CLS
50 PAPER 1 : WINDOW 2, 4, 1, 18 : CLS
60 PAPER 2 : WINDOW 5, 7, 1, 18 : CLS
70 PAPER 3 : WINDOW 8, 10, 1, 18 : CLS
80 PAPER 4 : WINDOW 11, 13, 1, 18 : CLS
90 PAPER 5 : WINDOW 14, 16, 1, 18 : CLS
100 PAPER 6 : WINDOW 17, 19, 1, 18 : CLS
110 PAPER 7 : WINDOW 2, 19, 19, 25 : CLS
120 GOTO 120
```

Comandă. Indică în mod text, dimensiunile unui canal de afişaj de pe ecran (în acest caz vorbim de fereastră). Vom supraveghea ca valorile parametrilor ⟨stînga⟩, ⟨dreapta⟩, ⟨sus⟩, ⟨jos⟩ să corespundă coordonatelor în vigoare în **MODE**-ecran utilizat. ⟨Numărul de canal⟩ va lua prin lipsă valoarea **#Ø**.

Cuvinte cheie asociate. **WINDOW SWAP**

WINDOW SWAP

WINDOW SWAP ⟨număr de canal⟩, ⟨număr de canal⟩

```
10 MODE 1 : INK 1, 24 : INK 2, 9 : INK 3, 6
```

```
20 WINDOW 21, 40, 13, 25 : PAPER 3
30 WINDOW #1, 1, 20, 1, 12 : PAPER #1, 2
40 CLS : PRINT #1, "Fereastra nr"
50 CLS # : PRINT #1, "Fereastra nr. 1"
60 LOCATE 1, 6
70 PRINT "Fereastră Roşie (0)"; SPC (2)
80 LOCATE #1, 1, 6
90 PRINT #1, "Fereastră verde (1)"
100 FOR t=1 TO 1000 : NEXT
110 WINDOW SWAP 0, 1 : GOTO 60
```

Comandă. Inversează prima şi a doua fereastră. Cele două ⟨numere de canal⟩ trebuie să figureze obligatoriu, fără ca ele să fie precedate, în acest caz precis, de indicatorul de canal #.

Această comandă permite dirijarea mesajelor BASIC pe un alt canal decît cel prin lipsă #∅.

Cuvinte cheie asociate. **WINDOW**

WRITE

WRITE [#⟨număr de canal⟩,] [⟨date de scris⟩]

```
10 REM scrie date pe dischetă
20 INPUT "dă-mi un număr"; a
30 INPUT "dă-mi un şir de caractere"; a$
40 OPENOUT "NOMFISH"
50 WRITE #9, a, a$
60 CLOSEOUT : PRINT "Datele sînt păstrate pe dischetă"
run
```

Comandă. Afişează sau scrie **(WRITE)** date pe canalul indicat. Două articole distincte trebuie să fie separate printr-o virgulă iar şirurile de caractere sînt aşezate între ghilimele. În exemplul nostru, datele de intrare se vor scrie pe canalul #9, adică înregistrate pe dischetă.

Pentru vizualizarea datelor, vom utiliza următorul program:

```
10 REM regăseşte datele pe dischetă
20 OPENIN "NOMFISH" : INPUT #9, a, a$
30 CLOSEIN : PRINT "cele două date sînt:"
40 PRINT : PRINT a, a$
```

Cuvinte cheie asociate. **INPUT, LINE INPUT**

XOR

⟨argument⟩ **XOR** ⟨argument⟩

```
IF "alin" ⟨"bebe" XOR "cîine"⟩ "pisică" THEN PRINT
"adevărat" ELSE PRINT "fals"
fals
PRINT 1 AND 1
0
```

PRINT 0 **AND** 0
0
PRINT 1 **AND** 0
1

Operator. Realizează bit cu bit operaţia booleeană **XOR** (SAU exclusiv) asupra unor numere întregi. Atunci cînd bits-ii celor două argumente nu sînt identici, bit-ul care rezultă are valoarea 1.

Cuvinte cheie asociate. **AND, OR, NOT**

XPOS

XPOS

```
10 MODE 1 : DRAW 320, 200
20 PRINT "Poziţia X a cursorului grafic=";
30 PRINT XPOS
```

Funcţiune. Indică, pe axa orizontală (X), poziţia cursorului grafic.

Cuvinte cheie asociate. **MOVE, MOVER, ORIGIN, YPOS**

YPOS

YPOS

```
10 MODE 1 : DRAW 320, 200
20 PRINT "Poziţia Y a cursorului grafic=";
30 PRINT YPOS
run
```

Funcţiune. Indică, pe axa verticală (Y), poziţia cursorului grafic.

Cuvinte cheie asociate. **MOVE, MOVER, ORIGIN, XPOS**

ZONE

ZONE ⟨număr intreg⟩

```
10 CLS : FOR z=2 TO 20
20 ZONE z
30 PRINT "X", "X ZONE="; z : NEXT
```

Comandă. Modifică mărimea tabulaţiei desemnate prin virgulă în comanda **PRINT.** Mărimea zonelor de afişaj (de 13 caractere prin lipsă) poate astfel să ia o valoare întreagă oarecare cuprinsă între 1 şi 255.

Cuvinte cheie asociate. **PRINT**

□ Utilizarea limbajului BASIC-AMSTRAD în carte

Comenzile şi instrucţiunile limbajului BASIC-AMSTRAD au fost utilizate în paginile: 19, 47, 92, 177, 241, 316, 374, 403, 492, 540, 546, 556, 580, 588, sintezele 15, 20, 21.

SINTEZA 17

Mai mult decît un memento al limbajului GW-BASIC de pe Felix PC (IBM PC) sub MS-DOS

□ Comenzi

AUTO

Generează automat un număr de linie la fiecare apăsare a tastei [CR] (RETURN).

BLOAD

Încarcă în memorie un fişier imagine a memoriei.

BSAVE

Salvează într-un fişier specificat de utilizator anumite secţiuni din memoria internă.

CHIDR

Modifică, catalogul curent.

CLEAR

Variabilele numerice iau valoarea zero, şirurile sînt iniţializate cu caracterul NUL, iar fişierele deschise sînt închise. Opţiunile suplimentare stabilesc adresa maximă de memorie accesibilă pentru GW–BASIC şi dimensiunea zonei de stivă.

CONT

Reia execuţia programului după **CTRL/BREAK** sau după execuţia enunţurilor **STOP** ori **END.**

DELETE

Şterge toate liniile programului curent.

EDIT

Permite modificarea unei anumite linii din program.

FILES

Afişează numele fişierelor aparţinînd catalogului indicat.

GWBASIC

Iniţializează mediul de operare şi al interpretorului GW–BASIC. *Este de fapt o comandă MS-DOS.*

KILL

Şterge un fişier pe disc.

LCOPY

Transferă conţinutul ecranului pe o imprimantă.

LIST

Listează textul programului curent pe ecran sau pe un fişier specificat în mod expres.

LLIST

Listează textul programului curent la imprimantă.

LOAD

Încarcă un program în memorie de pe dispozitivul specificat. Cu opţiunea R programul poate fi şi executat.

MERGE

Combină programul curent cu un fişier indicat în mod expres, care a fost anterior salvat în format ASCII.

MKDIR

Permite crearea unui nou catalog pe unitatea de disc specificată.

NAME

Modifică numele unui fişier de pe disc.

NEW

Şterge din memorie programul curent, inclusiv toate variabilele, permiţînd introducerea unui nou program.

RENUM

Modifică numerele liniilor programului curent.

RESET

Închide toate fişierele deschise de pe toate unităţile de disc.

RMDIR

Desfiinţează un catalog de pe discul specificat.

RUN

Comandă execuţia programului curent pe care eventual îl încarcă în prealabil de pe disc.

SAVE

Salvează programul curent pe disc, atribuindu-i un nume.

SHELL

Încarcă şi execută un alt program.

SYSTEM

Închide toate fişierele de date deschise, restituind controlul sistemului de operare (MS–DOS).

TRON

Provoacă listarea fiecărui număr de linie ce se execută.

TROFF

Anulează efectul comenzii **TRON.**

□ Instrucţiuni (enunţuri)

Enunţuri ce nu implică transfer de date

CALL [s]

Transferă controlul la o rutină în limbaj maşină.

CHAIN

Transferă controlul unui alt program BASIC, căruia îi transmite şi variabilele comune.

COM (n) {**ON** | **OFF** | **STOP**}

Activează **(ON)**, dezactivează **(OFF)**, suspendă **(STOP)** întreruperile de evenimente ale activităţii de COMunicaţii pe canalul n.

COMMON

Defineşte o zonă de date comune care nu este desfiinţată de către programul înlănţuit şi permite transferul variabilelor de la un program la altul.

DATE$

Stabileşte data curentă a zilei.

DEF FN

Defineşte şi stabileşte un nume pentru o funcţie utilizator.

DEF SEG

Atribuie adresa "segmentului" curent.

DEF USR

Activează accesul la o subrutină în limbaj maşină, specificînd adresa sa de început.

DEFINT, DEFSNG, DEFDBL, DEFSTR

Declară tipul variabilei în concordanţă cu sufixul (**INT**-întreg; **SNG**-simplă precizie; **DBL**-dublă precizie, **STR**-şir).

DIM

Specifică un nume de masiv, numărul de dimensiuni şi limitele superioare ale indicilor.

END

Termină execuţia unui program, închide fişierele deschise de date şi provoacă revenirea la nivel de comandă.

ENVIRON

Permite modificarea parametrilor tabelei de şiruri ale mediului GW–BASIC.

ERASE

Eliberează spaţiul şi numele de variabile rezervate în prealabil pentru masive (tablouri).

ERROR

Simulează apariţia unei erori GW–BASIC, sau generează o eroare definitivă de utilizator.

FOR/NEXT

Permite execuţia repetată, de un număr dat de ori, a unui grup de enunţuri.

GOSUB/RETURN

Se transferă controlul la linia indicată, considerată ca început al unei subrutine GW-BASIC (**GOSUB**) şi permite revenirea din aceasta la enunţul imediat următor apelului (**RETURN**).

GOTO

Transferă necondiţionat controlul unei linii specificate din programul curent.

IF . . . GOTO . . . ELSE

sau

IF . . . THEN . . . ELSE

Ia o decizie referitoare la fluxul programului curent, pe baza testării unei condiţii.

KEY { **OFF** | **ON** | **LIST** | n, ⟨expresieşir⟩ }

Atribuie cheii n şirul ⟨expresieşir⟩, activează (**ON**), dezactivează (**OFF**) listarea valorilor cheilor pe linia 25 a ecranului.

KEY (n) { **ON** | **OFF** | **STOP** }

Activează (**ON**), dezactivează (**OFF**), suspendă (**STOP**) întreruperile provocate de acţionarea cheii n.

LET

Atribuie o valoare unei variabile.

MID

Înlocuieşte un subşir cu altul.

ON COM(n) **GOSUB**

Specifică numărul primei linii al unei subrutine (asincrone) care va fi automat activată la apariţia unor caractere (nesolicitate) în zona tampon de comunicaţii.

ON ERROR GOTO

Activează întreruperea de eroare şi indică adresa primei linii a subrutinei ce va fi invocată în caz de eroare.

ON . . . GOSUB

Funcţie de condiţia specificată, apelează rutina GW-BASIC indicată prin numărul de ordine al primei sale linii.

ON . . . GOTO

Funcţie de condiţia specificată, provoacă transferul controlului la linia al cărui număr de ordine este specificat.

ON KEY(n) GOSUB

Specifică numărul de ordine al primei linii a unei subrutine ce va fi invocată cînd se acţionează cheia funcţională n.

ON PLAY(n) GOSUB

Specifică numărul de ordine al primei linii a unei subrutine ce va fi invocată cînd zona tampon muzicală conţine mai puţin de n note.

ON TIMER(n) GOSUB

Provoacă o întrerupere la fiecare n secunde.

OPTION BASE

Defineşte valoarea minimă a indicilor de tablouri.

PLAY

Execută o melodie în concordanţă cu un şir de note, precum şi modul lor de interpretare.

PLAY {ON | OFF | STOP}

Activare (**ON**), dezactivare (**OFF**), suspendare (**STOP**) întrerupere tip interpretare melodie.

POKE

Scrie un octet la o adresă de memorie.

RANDOMIZE

Restabileşte baza generatorului de numere aleatoare.

REM

Permite introducerea de comentarii explicative în program.

RESTORE

Permite ca enunţurile **DATA** să fie recitite fie de la începutul fişierului intern, fie de la o anume linie a sa.

RESUME

Continuă execuţia unui program după ce s-a executat o rutină de întrerupere ce tratează un caz de eroare.

STOP

Termină execuţia programului, revenindu-se la nivelul de comandă.

SWAP

Efectuează interschimbarea valorilor a două variabile.

TIME

Stabileşte valoarea orologiului curent.

TIMER {ON | OFF | STOP}

Activează (**ON**), dezactivează (**OFF**), suspendă (**STOP**) întreruperile de timp real.

WAIT

Suspendă execuţia programului în timpul coordonării stării unei porţi de intrare a maşinii.

WHILE/WEND

Provoacă execuţia iterativă a unui grup de enunţuri cît timp este adevărată o condiţie specificată.

Enunţuri ce implică transfer de date (Enunţuri de intrare/ieşire)

BEEP

Activează soneria (semnalul sonor).

CIRCLE(GR)*

Trasează un cerc (ori o elipsă) avînd un centru şi o rază specificate.

CLOSE

Termină operaţiile de intrare/ieşire cu un fişier ori un dispozitiv.

CLS

Şterge (curăţă) total sau parţial ecranul.

COLOR(MR)

Defineşte paleta de culori de fond şi active; de asemenea poate defini şi valorile implicite pentru acesta (inclusiv pentru texte).

---

* GR (grafic), MR (medie rezoluţie), HR (înaltă rezoluţie), SR (super rezoluţie), T (text), COM (comunicaţii), C (comandă), F (funcţie), S (enunţ), DC (comanda DOS).

COLOR(HR)
Definește culoarea curentă (implicită), de fond și culoarea pentru text.

COLOR(SR)
Definește culoarea curentă pentru desene, precum și culorile de fond și de text.

COLOR(T)
Stabilește culorile de fond și curente pentru text.

DATA
Crează un fișier intern de date aparținînd programului, ce va fi consultat de enunțurile **READ.**

DRAW(GR)
Trasează un obiect definit de o succesiune de comenzi de un singur caracter.

FIELD
Alocă spațiu pentru variabile într-o zonă tampon fișier în acces aleator.

GET(COM)
Citește un număr specificat de octeți într-o zonă tampon de comunicații.

GET(FIL)
Citește o înregistrare dintr-un fișier pe disc în acces aleator într-o zonă tampon cu acces aleator.

GET(GR)
Citește punctele dintr-o zonă a ecranului.

INPUT
Permite introducerea de date de la terminal în timpul execuției programului.

INPUT#
Citește articole de date de pe un fișier secvențial pe disc, atribuindu-le variabilelor programului.

IOCTL
Trimite un șir „Date de control" către un driver de dispozitiv în mod caracter, de fiecare dată după ce acesta a fost deschis.

LINE(GR)
Trasează o linie sau un dreptunghi.

LINE INPUT
Trasează o întreagă linie (pînă la 254 caractere) într-o variabilă tip șir, fără utilizarea delimitatorilor.

LINE INPUT#

Citeşte o întreagă linie (pînă la 254 caractere), fără delimitatori, de pe un fişier secvenţial pe disc, într-o variabilă de tip şir.

LOCATE(GR)

Deplasează cursorul în poziţia specificată; poate comuta cursorul (on/off) sau defini forma şi rata de clipire a cursorului.

LOCATE(T)

Deplasează cursorul în poziţia specificată a paginii active; poate comuta cursorul (on/off) şi defini dimensiunea cursorului utilizator (eventual şi a celui de suprascriere).

LPRINT

Tipăreşte datele la imprimantă.

LPRINT USING

Tipăreşte într-un anumit format, datele la imprimantă.

LSET/RSET

Transferă datele din memorie într-o zonă tampon a unui fişier în acces aleator.

ON TIMER(n) GOSUB

Provoacă o întrerupere la fiecare n secunde.

OPEN

Permite operaţii ulterioare de intrare/ieşire cu un fişier sau un dispozitiv.

OPEN COM

Deschide şi iniţializează un canal de comunicaţii pentru intrări/ieşiri.

OUT

Transmite un octet către un port de ieşire.

PAINT(GR)

Umple o zonă închisă de pe ecran cu o culoare specificată.

PRESET(GR)

Plasează un punct într-o poziţie specificată de pe ecran.

PRINT

Afişează datele pe ecran.

PRINT USING

Afişează, cu un format specificat, datele pe ecran.

PRINT#

Scrie datele secvenţial într-un fişier pe disc.

PRINT# USING

Scrie datele secvenţial într-un fişier pe disc, utilizînd un format specificat.

PSET(GR)

Desenează un punct într-o poziţie indicată pe ecran.

PUT(COM)

Scrie un număr indicat de octeţi într-un fişier de comunicaţii.

PUT(FIL)

Scrie o înregistrare dintr-o zonă tampon într-un fişier pe disc în acces aleator.

PUT(GR)

Transferă pe ecran o imagine grafică memorată într-un masiv (tablou).

READ

Citeşte valorile din unul sau mai multe enunţuri **DATA** şi le atribuie unor variabile.

RESET

Închide toate fişierele de date deschise de pe toate dispozitivele.

SCREEN

Stabileşte anumite caracteristici pentru ecranul dispozitivului display.

SOUND

Produce un semnal sonor via difuzor.

VIEW(GR)

Defineşte porţiuni de ecran ('vizoare') în care se vor proiecta conţinuturile ferestrelor.

VIEW PRINT

Stabileşte frontierele unei ferestre de text.

WIDTH

Stabileşte lăţimea în caractere a liniei.

WINDOW(GR)

Defineşte dimensiunile logice ale unui vizor.

WRITE

Scrie date pe ecran.

WRITE #

Scrie date într-un fişier secvenţial.

□ Funcţii

Funcţii numerice

ABS

Calculează valoarea absolută a unei expresii numerice.

ATN

Calculează arcul a cărui tangentă este argumentul.

CDBL

Converteşte o expresie numerică dată într-un număr real în dublă precizie.

CINT

Converteşte un argument numeric într-un întreg prin rotunjirea părţii fracţionare.

COS

Calculează cosinusul argumentului.

CSNG

Converteşte orice argument numeric într-un număr real în simplă precizie.

EXP

Calculează puterea de rangul argumentului a bazei logaritmilor naturali.

FIX

Calculează partea întreagă trunchiată a argumentului.

INT

Calculează cel mai mare întreg ce nu depăşeşte valoarea argumentului.

LOG

Calculează logaritmul natural al unui argument pozitiv.

RND

Determină un număr real aleator în intervalul 0–1.

SGN

Restituie valoarea 1 pentru un argument pozitiv, 0 pentru argument real şi –1 pentru argument negativ.

SIN

Calculează sinusul argumentului.

SQR

Calculează rădăcina pătrată dintr-o expresie numerică pozitivă.

TAN

Calculează tangenta unui argument.

Funcții referitoare la șiruri

ASC

Furnizează valoarea numerică a codului ASCII a primului caracter dintr-un șir.

{CVI | CVS | CVD}

Convertește valorile șir în valori numerice întregi (**CVI**), reale simplă precizie (**CVS**) ori dublă precizie (**CVD**).

INSTR

Caută prima apariție a unui subșir într-un șir, determinînd poziția în care are loc identificarea.

LEN

Calculează numărul de caractere dintr-un șir.

VAL

Convertește reprezentarea de tip șir a unui număr în valoarea sa numerică.

Funcții de Intrări/Ieșiri și diverse

CSRLIN

Calculează numărul liniei curente pe care se află cursorul.

EOF

Indică depistarea sfîrșitului unui fișier.

ERDEV

Reține valoarea efectivă a unei erori de dispozitiv.

ERR

Furnizează codul de eroare.

ERL

Furnizează numărul liniei unde s-a depistat eroarea.

FRE

Furnizează numărul de octeţi din memorie neutilizaţi încă de GW-BASIC.

INP

Furnizează octetul citit de la un port.

LOC

Furnizează poziţia curentă dintr-un fişier.

LOF

Determină numărul de octeţi alocaţi unui fişier.

LPOS

Determină poziţia curentă a capului de scriere într-o zonă tampon de tipărire.

PEEK

Furnizează octetul citit dintr-o adresă de memorie specificată.

PLAY

Furnizează numărul de note curente dintr-o zonă tampon muzicală.

PMAP(GR)

Aplică coordonatele fizice în cele univers sau invers.

POINT

Determină culoarea unui element de imagine de pe ecran, sau coordonata grafică curentă.

POS

Furnizează poziţia curentă a cursorului pe coloană.

SCREEN

Calculează codul ASCII (0–255) sau numărul culorii caracterului dintr-o poziţie specificată de pe ecran.

TIMER

Furnizează un număr în simplă precizie ce indică intervalul scurs de la miezul nopţii sau de la resetarea sistemului (în secunde).

USR

Apelează o subrutină în limbaj maşină.

VARPTR ({⟨variabilă⟩ | ⟨nume fis⟩})

Furnizează adresa unei variabile sau unei zone tampon intrare/ieşire disc (fişiere secvenţiale) ori zone tampon **FIELD** (fişiere aleatoare).

Funcţii de tip şir

Generale

CHR$

Furnizează un caracter avînd codul ASCII egal cu valoarea argumentului.

LEFT$

Extrage dintr-un şir subşirul stîng (prefix) avînd lungimea specificată.

MID$

Extrage dintr-un şir un subşir.

RIGHT$

Extrage dintr-un şir subşirul drept (sufix), avînd o lungime specificată.

SPACE$

Furnizează un şir constituit dintr-un număr specificat de spaţii.

STRING$

Furnizează un şir uniform de lungime dată avînd drept caracter de umplere caracterul iniţial al altui şir.

Funcţii de intrare/ieşire şi diverse

DATE$

Determină data curentă.

ENVIRON$

Permite regăsirea şirului mediu specificat din tabela de şiruri mediu aparţinînd sistemului GW-BASIC.

ERDEV$

Reţine numele dispozitivului tip caracter ce a provocat eroarea.

HEX$

Furnizează un şir reprezentînd valoarea hexazecimală a unui argument zecimal.

INKEY$

Restituie un şir de 1–2 caractere citite de la tastatură, sau şirul NULL dacă nici un caracter nu a fost citit de la tastatură.

INPUT$

Furnizează şirul de caractere citit de la tastatură sau dintr-un fişier.

IOCTL$

Furnizează un şir „Date de control" din driver-ul dispozitiv tip caracter ce este deschis.

{MKI$ | MKS$ | MKD$}

Converteşte valorile numerice tip întreg/simplă precizie/dublă precizie în valori de tip şir.

OCT$

Furnizează un şir ce reprezintă valoarea octală a unui argument zecimal.

SPC

Sare peste n spaţii în enunţurile **PRINT, LPRINT, PRINT#**.

STR$

Furnizează reprezentarea sub formă de şir a valorii unei expresii numerice specificate.

TAB

Mută cursorul sau capul de scriere în poziţia specificată în enunţurile **LPRINT, PRINT, PRINT#**.

TIME$

Furnizează orologiul curent.

VARPTR$

Furnizează formatul caractere a adresei de memorie a unei variabile.

☐ Cuvinte rezervate (definire, exemple – ordine alfabetică)

ABS[F]

ABS (⟨expresie numerică⟩)
OK
PRINT ABS (4 * (–5))
20
OK

Funcţie. Furnizează valoarea absolută pentru ⟨expresie numerică⟩.
Cuvinte cheie asociate. **SGN**

ASC[F]

ASC (⟨expresie tip şir⟩)
10 A$="EXEMPLU"
20 **PRINT ASC** (A$)
RUN
69
OK

Funcţie. Furnizează valoarea numerică a codului ASCII a primului caracter din ⟨expresie tip şir⟩.
Cuvinte cheie asociate. **CHR$**

ATN(F)

ATN (⟨expresie numerică⟩)
OK
PRINT ATN (1)
0.785398163
OK

Funcţie. Furnizează valoarea (în intervalul: $-\pi/2 \div \pi/2$) arcului a cărui tangentă este valoarea ⟨expresie numerică⟩. Valoarea este simplă precisă, exceptînd comutatorul /D la invocarea GW-BASIC.
Cuvinte cheie asociate. **COS, SIN, TAN**

AUTO(C)

AUTO [⟨număr linie⟩] [, ⟨increment⟩]
AUTO Generează numerele de linie 10, 20 30 . . .
AUTO 100, 25 Generează numerele 100, 125, 150 . . .
AUTO 300, Generează numerele 300, 325, 350 . . .
AUTO, 20 Generează numerele 0, 20, 40 . . .

Comandă. Se generează automat numere de linie începînd cu ⟨număr linie⟩ şi în paşii daţi de ⟨increment⟩. Valorile implicite sînt 10 (în lipsa argumentelor) sau valoarea ultimului ⟨increment⟩ precizat în comanda **AUTO** anterioară.
Cuvinte cheie asociate. **CTRL/BREAK** – revenire în mod comandă.

BEEP(S)

```
BEEP
  10 REM CALCUL RĂDĂCINA PĂTRATĂ
  20 INPUT X
  30 IF X>=0 THEN GOTO 50
  40 BEEP : GOTO 20
  50 PRINT SQR (X)
  RUN
  OK
```

Enunţ. Activează soneria.

BLOAD(C)

BLOAD (⟨specificator fis.⟩) [, ⟨ecart⟩]

```
10 REM ÎNCARCĂ UN PROGRAM ÎN LIMBAJ MAŞINĂ LA ADRESA
   60 : F000
20 DEF SEG 'RESTAUREAZĂ segmentul la segmentul de date al lui
   GW-BASIC
30 BLOAD "A : PROGAS", &HF000'ÎNCARCĂ PROGA ÎN segmentul
   de date GW-BASIC
```

Comandă. Încarcă în memorie programul în cod maşină care este memorat pe disc în fişierul ⟨specificator fiş⟩ şi care a fost salvat de o comandă **BSAVE** anterioară. Adresa de încărcare este chiar cea specificată în această din urmă comandă (în lipsa argumentului ⟨ecart⟩) sau deplasamentul ⟨ecart⟩

față de adresa segmentului definit de ultimul **DEF SEG** (în lipsă, adresa segmentului de date al interpretorului GW-BASIC).

Cuvinte cheie asociate. **BSAVE, CALL, DEF SEG**

BSAVE(C)

BSAVE (⟨specificator fiș⟩), ⟨ecart⟩, ⟨lungime⟩

```
10 REM Salvare 512 octeți începînd cu 6000 : F000 în fișier PROGA
20 DEF SEG=&H6000
30 BSAVE "PROGA", &HF000, 512
```

Comandă. Salvează octet cu octet orice porțiune de memorie de ⟨lungime⟩ octeți (fie rutină mașină, fie zonă tampon ecran etc.), care începe de la un deplasament ⟨ecart⟩ față de segmentul definit de un **DEF SEG** anterior (implicit segmentul de date al interpretorului GW-BASIC), în fișierul disc ⟨specificator fiș⟩.

Cuvinte cheie asociate. **BLOAD, CALL, DEF SEG**

CALL(S)

CALL ⟨var num⟩ [(⟨listă de variabile⟩)]

```
200 RUTEXT=&HD000
210 CALL RUTEXT (I, J).
```

Enunț. Se transferă controlul unei subrutine externe care este prezentă în memorie la adresa dată de deplasamentul ⟨varnum⟩ față de segmentul definit de ultimul **DEF SEG** (implicit segmentul de date al interpretorului GW-BASIC).

Opțional se pot transmite argumentele date în ⟨listă de variabile⟩.

Cuvinte cheie asociate. **BLOAD, CALLS, DEF SEG, USR**

CALLS(S)

CALLS ⟨varnum⟩ [(⟨lista de variabile⟩)]

Enunț. Se transferă controlul unei subrutine externe aflate la adresa relativă ⟨varnum⟩ față de segmentul curent, opțional transmițîndu-se și argumentele indicate în ⟨listă de variabile⟩, sub formă de adrese segmentate.

Cuvinte cheie asociate. **BLOAD, CALL, DEF SEG**

CDBL(F)

CDBL ⟨expresie numerică⟩

```
10 UNGHI=454.67
20 PRINT UNGHI; CDBL(UNGHI)
RUN
   454.67      454.6700134277344
```

Funcție. Convertește ⟨expresie numerică⟩ din simplă precizie în dublă precizie.

CHAIN(S)

CHAIN [MERGE] (⟨specificator fișier⟩), [, [⟨număr linie⟩] [, [**ALL**] [, **DELETE** ⟨domeniu⟩]]]

```
110 REM ÎNLĂNȚUIRE CU TRANSMITERE ARGUMENTE ÎN ZONE
    COMUNE
120 REM PROGRAMUL E SALVAT ÎN FIȘIERUL "PROGRA"
130 DIM X$(2), Y$(2)
```

```
140 COMMON X( ), Y( )
150 X$(1)="ÎNAINTEA ÎNLĂNŢUIRII SE ATRIBUIE VALORI"
160 X$(2)="VARIABILELOR COMUNE"
170 Y$(1)=" " : Y$(2)=" "
180 CHAIN "PROGRB"
190 PRINT : PRINT Y$(1) : PRINT : PRINT Y$(2) : PRINT
200 END
110 REM ENUNŢUL "DIM X$(2), Y$(2)" SE EXECUTĂ O SINGURĂ
    DATĂ
120 REM DECI NU MAI APARE ŞI ÎN ACEST PROGRAM
130 REM PROGRAMUL E SALVAT ÎN FIŞIERUL "PROGRB"
140 COMMON X( ), Y( )
150 PRINT : PRINT X$(1); X$(2)
160 Y$(1)="SPECIFICIND LINIA DE UNDE ÎNCEPE PROGRA"
170 Y$(2)="NU MAI TREBUIE EXECUTAT ENUNŢUL DE DIMENSIO
    NARE"
180 CHAIN "PROGRA", 190
190 END
```

Enunţ. Transferă controlul unui program memorat pe disc în fişierul ⟨specificator fişier⟩ de la început, sau opţional de la linia avînd numărul dat de expresia ⟨număr linie⟩ şi din care s-au eliminat liniile din ⟨domeniu⟩. Opţiunea **MERGE** efectiv actualizează programul curent, păstrînd deschise fişierele şi conservînd variabilele şi funcţiile definite de utilizator. Opţiunea **ALL** transferă toate variabilele din programul curent în programul ţintă.

CHDIR(C)

CHDIR ⟨nume traseu⟩

```
10 REM SUBCATALOGUL CATAL2 AL CATALOGULUI CATAL1 DIN
   RĂDĂCINĂ
15 REM DEVINE CATALOGUL CURENT PE DISCUL A
20 CHDIR "A : \ CATAL 1 \ CATAL 2"
```

Comandă. Catalogul curent devine cel din capătul traseului ⟨nume traseu⟩ (expresie tip şir).

Cuvinte cheie asociate. **MKDIR, RMDIR**

CHR$(F)

CHR$ (⟨număr întreg⟩)

```
30 REM DECODIFICA ALFABETUL
40 INPUT I%
50 K%=I%+64
60 PRINT CHR$ (K%)
RUN
```

Funcţie. Furnizează un şir de un caracter al cărui cod zecimal ASCII este valoarea argumentului numeric ⟨întreg⟩. Utilizat mai ales pentru construire de caractere speciale (netipăribile).

Cuvinte cheie asociate. **ASC**

CINT(F)

CINT (⟨expresie numerică⟩)

OK

```
PRINT CINT (83.27)
  83
PRINT CINT (17.536)
  18
OK
```

Funcţie. Converteşte ⟨expresie numerică⟩ într-o valoare întreagă cu rotunjirea părţii fracţionare.

Cuvinte cheie asociate. **CDBL, CSNG, FIX, INT**

CIRCLE(SGR)

CIRCLE [**STEP**] (⟨abscisă⟩, ⟨ordonată⟩), ⟨rază⟩, [, ⟨culoare⟩ [, ⟨start⟩, ⟨end⟩ [, ⟨aspect⟩]]]

```
10 SCREEN 1
20 COLOR 0, 0, 3, 0
30 CLS
40 CIRCLE (100, 120), 90
50 CIRCLE (150, 130), 120
60 CIRCLE (250, 120), 100
70 PAINT (180, 120)
```

Enunţ. Trasează un (arc de) cerc sau o elipsă cu centrul în punctul de coordonate ⟨abscisă⟩, ⟨ordonată⟩ şi de rază ⟨raza⟩, calculate în elemente de imagine (pixeli), eventual între unghiurile ⟨start⟩ şi ⟨end⟩ (exprimate în radiani în intervalul: $-2\pi \div +2\pi$) cu culoarea ⟨culoare⟩ (expresie întreagă 0–3) şi cu un raport de aspect ⟨aspect⟩ (implicit 5/6). Prin convenţie un unghi negativ înseamnă şi trasarea razei. Coordonatele sînt absolute sau relative (în opţiunea **STEP**).

CLEAR(C)

CLEAR [,[⟨memorie⟩][,⟨stivă⟩]]

```
CLEAR
CLEAR , 32768
CLEAR , , 2000
CLEAR , 32768, 2000
```

Comandă. Repune pe zero toate variabilele numerice, iniţializează cu caracterul **NULL** toate variabilele de tip şir, închide toate fişierele deschise, eliberează zonele tampon disc, anulează variabilele comune. Opţional se poate preciza şi extensia maximă a segmentului de date GW–BASIC la valoarea ⟨memorie⟩, precum şi spaţiul organizat ca stivă (implicit min (128, ⟨memorie⟩/8).

Cuvinte cheie asociate. **DEF FN, DEFINT / SNG / DBL / STR, DEF SEG, DEF USR**

CLOSE

CLOSE ⟨listă numere canale⟩
⟨listă numere canale⟩ : : =[#]⟨nr. canal⟩ | ⟨listă numere canale⟩,[#]⟨nr.canal⟩

```
100 OPEN "I", #2, "FISDAT" ' DESCHIDE FIŞIER DATE
```

.
.
.
320 **CLOSE** #2 ' ÎNCHIDE FIŞIERUL

Enunţ. Termină operaţiile de intrare/ieşire referitoare la fişierul sau dispozitivul GW-BASIC asociate fiecăreia din expresiile întregi ⟨nr. canal⟩ (asociere efectuată de un enunţ **OPEN** anterior). Enunţul **END** şi comanda **NEW** efectuează implicit această acţiune pentru toate fişierele deschise.

Cuvinte cheie asociate. **END, NEW, OPEN**

CLS (S)

CLS [⟨n⟩]

10 **CLS** ' Şterge ecranul sau umple cu culoarea de fond vizorul grafic sau fereastra de text
35 **CLS** 0 ' Şterge tot ecranul
75 **CLS** 1 ' Vizorul grafic umplut cu culoarea de fond
140 **CLS** 2 ' Fereastra de text umplută cu culoarea de fond

Enunţ. Efectuează umplerea cu culoarea fondului a întregului ecran (⟨n⟩=0), a vizorului grafic (⟨n⟩=1) ori a ferestrei de text (⟨n⟩=2). În lipsa argumentelor are efectul implicit al situaţiei curente, în ordinea fereastră text, vizor grafic, tot ecranul, restabilind afişarea liniei cu chei funcţionale.

Cuvinte cheie asociate. **SCREEN, VIEW, WIDTH**

COLOR (ŞMR)

COLOR [⟨fond⟩][,⟨paletă⟩][,⟨color grafic⟩][,[⟨fond grafic⟩]] [,⟨color text⟩]]]

10 **SCREEN** 1, 0
15 **REM** FONDUL ESTE VERDE DESCHIS, PALETA 1 (CIAN, MAGENTA, ALB)
20 **REM** DESENELE SE TRASEAZĂ CU MAGENTA PE FOND NEGRU
25 **COLOR** 10, 1, 2, 0

Enunţ. Defineşte un ⟨fond⟩ (expresie numerică 0–31, modulo 15) pentru caractere, o ⟨paletă⟩ coloristică (expresie numerică 0–255 modulo 2 pentru detalierea culorilor 1, 2, 3 : verde-cian, roşu-magenta, galben-alb), culoarea desenelor ⟨color grafic⟩ (expresie numerică 0–3), culoarea caracterelor ⟨color text⟩ (expresie numerică 0–7) şi culoarea de fond ⟨fond grafic⟩ (expresie numerică 0–3) pentru vizorul grafic.

Valori implicite: ⟨fond⟩=0, ⟨paletă⟩=1, ⟨fond grafic⟩=0, ⟨color grafic⟩=3, ⟨color text⟩=3.

Cuvinte cheie asociate. **SCREEN**

COLOR (SHR)

COLOR [⟨color grafic⟩][,⟨fond grafic⟩][,⟨color text⟩]

10 **SCREEN** 2
15 **REM** DESENARE CU NEGRU PE FOND ALB
20 **COLOR** 0, 1, 0

Enunţ. Defineşte culoarea implicită a desenelor ⟨color grafic⟩ (expresie întreagă 0–3 modulo 2; implicit 1), a caracterelor text ⟨color text⟩ (expresie numerică întreagă care se pune în operaţie **XOR** cu harta alocare ecran) pe un ⟨fond grafic⟩ (expresie numerică 0–1).

Cuvinte cheie asociate. **SCREEN**

COLOR (SSR)

COLOR [⟨color grafic⟩][,⟨fond grafic⟩][,⟨color text⟩]

```
10 SCREEN 3
15 REM DESENEAZĂ CU NEGRU PE ALB CU VIDEO INVERS
20 REM PENTRU TEXT
25 COLOR 0, 1, 0
```

Enunț. Definește culoarea implicită a desenelor ⟨color grafic⟩ (expresie întreagă 0–3 modulo 2; implicit 1), a caracterelor alfanumerice ⟨color text⟩ (expresie numerică: 0 – video normal; 1 – video invers; >1 operație **XOR** cu harta alocare ecran) pe un ⟨fond grafic⟩ (expresie numerică 0–1; implicit 0).

Cuvinte cheie asociate. **SCREEN**

COLOR (ST)

COLOR [⟨color⟩][,⟨fond⟩][,⟨vid⟩]

```
 90 REM SE VA SCRIE CU NEGRU PE FOND VERDE
100 COLOR 0, 2
```

Enunț. Stabilește culoarea ⟨color⟩ (expresie numerică întreagă 0–31) pe un ⟨fond⟩ (expresie numerică 0–15 modulo 8). Argumentul ⟨vid⟩ este pentru compatibilitate cu alte sisteme BASIC. Codurile sînt 0 – alb, 1 – albastru; 2 – verde; 3 – cian.; 4 – roșu; 5 – magenta; 6 – maron; 7 – alb; 8 – gris; 9 – albastru aprins; 10 – verde aprins; 11 – cian aprins; 12 – roșu aprins; 13 – magenta aprins; 14 – galben; 15 – alb intens. Adăugarea culorii 16 la aceste culori provoacă clipirea culorii de bază.

COM (S)

COM (⟨număr canal⟩){**ON** | **OFF** | **STOP**}

```
 5 REM ACTIVARE ÎNTRERUPERE COMUNICAȚIE PE CANAL 2
10 COM (2) ON
```

Enunț. În varianta **ON** se activează întreruperea datorată unui eveniment pe canalul ⟨număr canal⟩ (expresie întreagă 1–4).

În varianta **OFF** întreruperea se dezactivează, iar în varianta **STOP** se suspendă (amînă).

Cuvinte cheie asociate. **ON COM** (⟨nr. canal⟩) **GOSUB** ⟨nr. linie⟩

COMMON (S)

COMMON ⟨listă de variabile⟩

```
10 REM PROG1
20 DEF DBL C
30 DIM A%(2)
40 COMMON A%( ), B%, C
50 A%(1)=5
60 A%(2)=17
70 C=3.145967
80 B%=23
90 CHAIN "PROG2"
100 END
```

```
10 REM PROG2
20 DEFDBLC
30 PRINT A%; B%; C
40 END
```

Enunț. Variabilele numerice sau de tip șir din ⟨listă de variabile⟩ sînt plasate într-o arie comună, fiind accesibile tuturor programelor GW-BASIC înlănțuite prin enunțul **CHAIN**. Variabilele tip masiv sînt specificate prin grupul ' ( ) '. Declarația are loc numai în programul apelant, care obliga-

toriu le va şi iniţializa dar în programul apelat se vor insera eventualele enunţuri de definire explicită a tipului (**DEFINT, DEFSNG, DEFDBL, DEFSTR**).

CONT (C)

CONT

```
10 INPUT X, Y                RUN              CONT
20 PRODUS=X*Y                ?18, 2.5         165.5
30 STOP                      Break in 30      OK
40 PRINT PRODUS+120.5        OK
.                            PRINT PRODUS
                             45
                             OK
```

Comandă. Reia execuţia programului GW-BASIC după ce s-a tastat **CTRL/BREAK** ori s-au executat enunţurile **STOP** sau **END**, din punctul de întrerupere.

Cuvinte cheie asociate. **END, STOP**

COS (F)

COS (⟨expresie numerică⟩)

```
10 A=.4
20 PRINT COS(A)
RUN
.921061
OK
```

Funcţie. Calculează cosinusul argumentului determinat de valoarea în radiani a expresiei numerice ⟨expresie numerică⟩.

Cuvinte cheie asociate. **SIN**

CSNG (F)

CSNG (⟨expresie numerică⟩)

```
10 A#=454.6700134277344
20 PRINT A#; CSNG (A#)
RUN
 454.6700134277344  454.67
OK
```

Funcţie. Converteşte valoarea argumentului numeric ⟨expresie numerică⟩ într-un număr simplă precizie.

Cuvinte cheie asociate. **CDBL, CINT**

CSRLIN (F)

CSRLIN

```
10 REM SE VA DETERMINA POZIŢIA CURENTĂ A CURSORULUI
12 REM SE VA SCRIE UN MESAJ PE PRIMA LINIE
15 REM SE VA RESTAURA APOI CURSORUL
20 X=CSRLIN
30 Y=POS (0)
40 LOCATE 1, 1
50 PRINT "Terminalul este la dispoziţia dumneavoastră"
60 LOCATE X, Y
```

Funcție. Furnizează o valoare întreagă în domeniul 1–25 reprezentînd numărul liniei pe care se află cursorul.

Cuvinte cheie asociate. **LOCATE, POS.**

CVI (F)

CVI (⟨char 2⟩)
100 **FIELD**#1, 2 **AS X$**
110 **GET**#1
120 X%=CVI(X$)

Funcție. Convertește șirul de două caractere numerice ⟨char 2⟩ într-o valoare întreagă.

Cuvinte cheie asociate. **MKI, MKD, MKS, CVD, CVS.**

CVD (F)

CVD (⟨char 8⟩)
100 **FIELD**#1, 8 **AS X$**
110 **GET**#1
120 X# =**CVD** (X$)

Funcție. Convertește șirul de opt caractere numerice ⟨char 8⟩ într-un număr real în dublă precizie.

Cuvinte cheie asociate. **MKI, MKD, MKS, CVI, CVS.**

CVS (F)

CVS (⟨char 4⟩)
100 **FIELD**#1, 4 **AS X$**
110 **GET**#1
120 X=**CVS** (X$)

Funcție. Convertește șirul de patru caractere numerice ⟨char 4⟩ într-un număr real în simplă precizie.

Cuvinte cheie asociate. **MKI, MKD, MKS, CVI, CVS.**

DATA (S)

DATA ⟨lista de constante⟩
OK
10 **PRINT** "NUME", "PRENUME", "VIRSTA"
20 **READ** A$, B$, V%
30 **DATA** "DUMITRAȘCU,", "LIVIU", 40
40 **PRINT** A$, B$, V%
RUN
NUME PRENUME VIRSTA
DUMITRAȘCU, LIVIU 40

Enunț. Creează (sau extinde) un 'fișier intern' format dintr-un șir de constante specificate în ⟨lista de constante⟩. Articolele din fișier sînt accesibile prin enunțuri **READ**. Fișierul poate fi repoziționat la început cu enunțul **RESTORE.**

Cuvinte cheie asociate. **READ, RESTORE.**

DATE$ (F)

<varsir>=**DATE$**
PRINT DATE$
01–26–1988

Funcţie. Obţine data zilei din sistem şi o restituie în variabila şir <varsir> sub forma unui şir de 10 caractere în forma ll–zz–aaaa cu ll=01–12 (luna), zz=01–31 (ziua) şi aaaa=1980–2099 (anul)

DATE$ (S)

DATE$=<expresie şir>
DATE$="7–02–5"
PRINT DATE$
07–02–2005

Enunţ. Expresia tip şir <expresie şir>, trebuie să fie de forma "<lun>–<zi>–<an>" sau "<lun>/<zi>/<an>", unde <lun> este un şir de 1–2 cifre semnificînd luna, <zi> un şir de 1–2 cifre semnificînd ziua (cînd este o cifră, automat se presupune că e precedată de caracterul 0). Pentru <an> se admit o cifră (care se adaugă la valoarea 2000), 2 cifre (ce se adaugă la valoarea 1900) sau patru cifre.

DEF FN (S)

DEF FN <numef> [(<listă argumente formale>)]=<expresie>
10 **DEF FN TANG** (X)=**SIN** (X)/**COS** (X)
20 A= . 785395
30 **PRINT TANG** (A)
1

Enunţ. Defineşte funcţia utilizator <numef> care are lista de argumente formale <listă argumente formale> şi are ca valoare expresia <expresie>. Trebuie să existe concordanţă între tipul <expresie> şi tipul <numef>.

DEF SEG (S)

DEF SEG [=<adresă>]
10 **DEF SEG**=&HB800 ' Segmentul are adresa tamponului de ecran
20 **DEF SEG** ' Se restabileşte segmentul la adresa segmentului de date GW-BASIC

Enunţ. Adresa segmentului curent GW-BASIC este la valoarea 16* <adresă> (expresie numerică întreagă 0–65535); valoarea implicită este segmentul de date GW-BASIC.

Cuvinte cheie asociate. **BLOAD, BSAVE, CALL, DEF USR, PEEK, POKE.**

DEF USR (S)

DEF USR [<n>]=<ecart>
120 **DEF SEG**=0
190 **DEF USRO**=24000
200 X=**USRO** (Y∧2/2.93)

Enunţ. Dă posibilitatea accesului la o subrutină în limbaj maşină numită **USR** <n> unde n ia valoarea 0–9, adresa sa de început fiind privită

ca o deplasare ⟨ecart⟩ (expresie numerică fără semn 0–65535) față de segmentul curent (definit cu **DEF SEG**).

Cuvinte cheie asociate. **USR**.

DEFINT (S)

DEFINT ⟨listinit⟩
cu
⟨listinit⟩ : : =⟨init⟩ | ⟨listinit⟩, ⟨init⟩
⟨init⟩ : : =⟨literă⟩ | ⟨literă 1⟩–⟨literă 2⟩.
10 **REM** TOATE VARIABILELE AL CĂROR NUME ÎNCEPE CU I, J, K
L, U, V, W, X, Y SÎNT ÎNTREGI.
15 **DEFINT** I–L, U–Y

Enunț. Declară ca întreg tipul variabilelor al căror nume începe cu litera ⟨literă⟩ sau orice literă din intervalul ⟨literă 1⟩–⟨literă 2⟩.

Cuvinte cheie asociate. **DEFDBL, DEFSNG, DEFSTR.**

DEFDBL (S)

DEFDBL ⟨listinit⟩
cu
⟨listinit⟩ : : =⟨init⟩ | ⟨listinit⟩, ⟨init⟩
⟨init⟩ : : =⟨litera⟩ | ⟨literă 1⟩–⟨literă 2⟩
10 **REM** DEFINEȘTE CA REALE DUBLĂ PRECIZIE VARIABILELE AL
CĂROR NUME ÎNCEPE CU A, B, C
15 **DEFDBL** A–C

Enunț. Declară ca real dublă precizie tipul variabilelor al căror nume începe cu litera ⟨litera⟩ ori cu oricare literă din intervalul ⟨literă 1⟩–⟨literă 2⟩.

Cuvinte cheie asociate. **DEFINT, DEFSNG, DEFSTR**

DEFSNG (S)

DEFSNG ⟨listinit⟩
cu
⟨listinit⟩ : : =⟨init⟩ | ⟨listinit⟩, ⟨init⟩
⟨init⟩ : : =⟨litera⟩ | ⟨literă 1⟩–⟨literă 2⟩
10 ' DEFINEȘTE CA REALE SIMPLĂ PRECIZIE VARIABILELE AL CĂROR
NUME ÎNCEP CU D, F, G, H
20 **DEFSNG** D, F–H

Enunț. Declară ca real simplă precizie tipul variabilelor al căror nume începe cu litera ⟨litera⟩ ori cu orice literă din intervalul ⟨litera1⟩–⟨litera2⟩.

Cuvinte cheie asociate. **DEFINT, DEFDBL, DEFSTR.**

DEFSTR (S)

DEFSTR ⟨listinit⟩
cu
⟨listinit⟩ : : =⟨init⟩ | ⟨listinit⟩, ⟨init⟩
⟨init⟩ : : = ⟨litera⟩ | ⟨litera 1⟩–⟨litera 2⟩
⟨literă1⟩–⟨literă2⟩
10 **REM** DEFINEȘTE CA TIP ȘIR VARIABILELE AL CĂROR NUME ÎN-
CEPE CU M, N, O, P, Q, T
20 **DEFSTR** M–Q, T

Enunţ. Declară ca şir tipul variabilelor al căror nume începe cu litere ⟨litera⟩ ori cu orice literă în intervalul ⟨literă1⟩–⟨literă2⟩.

Cuvinte cheie asociate. **DEFINT, DEFDBL, DEF SNG.**

DELETE (C)

DELETE ⟨interlin⟩
cu
⟨interlin⟩ : : =⟨număr linie⟩ | ⟨număr linie1⟩–⟨număr linie2⟩

```
DELETE  -60       ' ELIMINAREA TUTUROR LINIILOR
                    PÎNĂ LA 60 INCLUSIV
DELETE  75        ' ELIMINAREA LINIEI 75
DELETE  120-130   ' ELIMINAREA LINIILOR ÎNTRE 120
                    şi 130 INCLUSIV
DELETE  165-      ' ELIMINAREA TUTUROR LINIILOR
                    DE LA 165 PÎNĂ LA SFIRŞIT PROGRAM.
```

Enunţ. Eliminarea dintr-un program GW-BASIC a liniei avînd numărul ⟨număr linie⟩ sau a liniilor avînd numele cuprinse între valorile expresiilor întregi ⟨număr linie1⟩ şi ⟨număr linie2⟩. Linia curentă este referită cu '.'.

DIM (S)

DIM ⟨listă de tablouri⟩
cu
⟨tablou⟩ : : =⟨nume tablou⟩ | (⟨listă limite⟩)

```
10 REM UN TABLOU UNIDIMENSIONAL CU INDICE 1-20 E
   INIŢIALIZAT CU PĂTRATELE RANGULUI
20 OPTION BASE 1
30 DIM A% (10)
40 FOR I%=0 TO 10
50    A%(I%)=I%^2
60 NEXT I%
```

Enunţ. Specifică un tablou de elemente avînd numele ⟨nume tablou⟩, dimensiunea dată de numărul componenţilor din ⟨listă limite⟩, fiecare dimensiune avînd o ⟨limită⟩ (expresie numerică întreagă) superioară. Tipul tabloului este dat de utilizarea convenţiilor generale (eventual combinate cu declaraţii implicite gen **DEFINT, DEFDBL, DEFSNG, DEFSTR**). Limita inferioară a indicilor este dată de un enunţ **OPTION BASE** anterior (explicit sau implicit).

Elementele numerice sînt iniţializate cu 0, elementele şir cu caracterul **NULL.**

Cuvinte cheie asociate. **DEFINT, DEFDBL, DEFSNG, DEFSTR, OPTION BASE.**

DRAW (S)

DRAW ⟨expresie şir⟩

```
10 REM TRASARE PĂTRAT
20 SCREEN 1
30 X=30
40 DRAW "U=X; R=X; D=X; L=X";
10 REM TRASEAZĂ UN TRIUNGHI
```

20 **SCREEN** 1
30 **DRAW** "BM10, 10M35, 60M60, 5M10, 10".

Enunț. Se trasează o imagine specificată de comenzile GML aparținînd expresiei șir ⟨expresie șir⟩.

Comenzile limbajului GML sînt:

U[⟨n⟩] – Deplasare spot în sus cu ⟨n⟩ * ⟨factor scară⟩ puncte;
D[⟨n⟩] – Deplasare spot în jos cu ⟨n⟩ * ⟨factor scară⟩ puncte;
L[⟨n⟩] – Deplasare spot la stînga cu ⟨n⟩ * ⟨factor scară⟩ puncte;
R[⟨n⟩] – Deplasare spot la dreapta cu ⟨n⟩ * ⟨factor scară⟩ puncte;
E[⟨n⟩] – Deplasare spot SV–NE cu ⟨n⟩ * ⟨factor scară⟩ puncte;
M⟨x⟩, ⟨y⟩ – Deplasare spot într-o poziție care este exprimată absolut dacă expresiile numerice întregi ⟨x⟩ și ⟨y⟩ nu sînt precedate de semnul plus (+) sau (–), ori relativ în caz contrar;
B – Deplasare fără trasare;
N – Deplasare în poziția inițială;
A⟨n⟩ – Rotarea figurii cu ⟨n⟩ * 90 grade, unde ⟨n⟩ este o expresie numerică întreagă (0–3);
TA⟨n⟩ – Rotarea figurii cu ⟨n⟩ grade, unde ⟨n⟩ este expresie numerică întreagă (–360 ÷ +360);
C⟨n⟩ – Stabilirea culorii la valoarea ⟨n⟩; (expresie numerică întreagă 0–3 pentru rezoluție medie, 0–1 pentru rezoluție mare și foarte mare);
S⟨k⟩ – Stabilește un factor de scară egal cu 1/4 din ⟨k⟩ (expresie numerică întreagă 1–255);
P⟨n⟩, ⟨m⟩ – Alegerea unei culori ⟨n⟩ pentru colorarea interiorului unei figuri și a unei culori ⟨m⟩ pentru contur. Argumentele ⟨n⟩ și ⟨m⟩ sînt expresii numerice întregi (0–3 pentru rezoluție medie; 0–1 pentru rezoluție mare și foarte mare);
X⟨exprsubșir⟩ – Executarea comenzilor GML din subșirul ⟨exprsubșir⟩

În limbajul GML există următoarele exprimări:
⟨comandă GML⟩ ⟨valoare numerică⟩ [;]
⟨comandă GML⟩ = ⟨nume variabilă⟩;

Cuvinte cheie asociate. **SCREEN**

EDIT(C)

EDIT ⟨număr linie⟩
EDIT 310
310 **LET** A=B * C
↑

Comandă. Permite modificarea (în mod imediat) a liniei ⟨număr linie⟩ (dacă e linia curentă se notează cu '.', altfel e o expresie numerică întreagă), utilizînd cheile funcționale de editare.

END(S)

END
110 **IF** K%<5 **GOTO** 20 **ELSE END**

Comandă. Semnalează terminarea execuției programului și revenirea în GW–BASIC la nivel de comandă.

ENVIRON(S)

ENVIRON ⟨param⟩
```
10 REM MODIFICAREA TRASEULUI CURENT DE ACCES LA FIŞIERE
20 ENVIRON "PATH=A : SERVICIU; A : COLECTIV"
```
Enunţ. Se modifică tabela şirurilor de definire a mediului GW–BASIC potrivit expresiei şir ⟨param⟩ care e de forma:

a) ⟨identificator parametru⟩=⟨text⟩ (pentru modificarea valorii lui ⟨identificator parametru⟩ cu valoarea ⟨text⟩);

b) ⟨identificator parametru⟩=; (pentru eliminarea ⟨identificator parametru⟩ din tabelă).

Cuvinte cheie asociate. **ENVIRON$**

ENVIRON$(F)

ENVIRON$ (⟨identificator parametru⟩) | **ENVIRON** (⟨n⟩)
```
10 A$=ENVIRON$ ("PATH")
```
Funcţie. Regăsirea valorilor lui ⟨identificator parametru⟩ sau al parametrului de rang ⟨n⟩ (expresie numerică întreagă 1–255).

Cuvinte cheie asociate. **ENVIRON**

EOF(F)

EOF (⟨număr fişier⟩)
```
10 REM CITIRE ÎN MASIV M A VALORILOR DIN FIŞIERUL "STOC"
20 OPTION BASE 1
30 DIM M (100)
40 OPEN "I", 2, "STOC"
50 I=1
60 IF EOF (2) THEN 250
70 INPUT #2, M(I)
30 I=I+1
90 GOTO 60
```
Funcţie. Returnează valoarea "adevărat", dacă s-a depistat condiţia de 'sfîrşit de fişier' pentru fişierul ⟨număr fişier⟩ (expresie numerică întreagă), desemnînd un fişier definit într-un enunţ **OPEN** anterior). Condiţia de sfîrşit de fişier este îndeplinită după cum urmează:

– fişiere secvenţiale : tentativă de citire după marca de fişier;
– fişiere aleatoare : tentativă de citire după sfîrşitul logic;
– fişiere de comunicaţii ASCII : caracter CTRL/Z citit;
– fişiere de comunicaţii binare : coada de intrare vidă.

Cuvinte cheie asociate. **LOC, OPEN**

ERASE(S)

ERASE ⟨listă masive⟩
```
10 REM ELIBEREAZĂ SPAŢIUL REZERVAT MASIVELOR X, Y ÎN SCO
   PUL REUTILIZĂRII LA REDIMENSIONARE
20 DIM X(20, 30), Y(45)
   .
   .
   .
200 ERASE X, Y
210 DIM X(100), Y(10, 30, 5)
```

Enunţ. Eliberează spaţiul utilizat pentru numele masivelor din ⟨listă masive⟩ ca şi pentru elementele acestora.

ERDEV(F)

ERDEV

Funcţie. Furnizează o valoare întreagă ce conţine codul de eroare depistat de ultimul dispozitiv ce a semnalat o eroare. Biţii superiori (13–15) sînt biţi atribute cuvînt din blocul de antet al dispozitivului, iar biţii inferiori (0–7) codul de întrerupere X'24' MS–DOS.

ERDEV$(F)

ERDEV$

Funcţie. Furnizează un şir de 8 caractere ce reprezintă numele dispozitivului caracter la care s-a semnalat ultima eroare, sau un şir de 2 caractere ce reprezintă numele blocului nume dispozitiv (dacă dispozitivul nu e de tip caracter, ci de exemplu disc).

ERL(F)

```
ERL
 10 ON ERROR GOTO 100
 20 INPUT "Ziua curentă"; Z%
 30 IF (Z%<0) OR (Z%>3 1) THEN ERROR 250
 – – – – – – – – – – –
100 IF (ERR>76) THEN PRINT "Zi imposibilă"; Z%; ERR; ERL
110 RESUME 20
120 END
RUN
Ziua curentă ? 45
Zi imposibilă 45 75 30
Ziua curentă ? 28
.
.
.
```

Funcţie. Restituie o valoare întreagă reprezentînd numărul liniei la care s-a depistat eroarea, neputînd deci fi citată corect în modul direct. De obicei e utilizată sub forma:

IF ERL = ⟨număr linie⟩ **THEN** . . .

Cuvinte cheie asociate. **ERR, ERROR, ON ERROR**

ERR(F)

ERR

Funcţie. Restituie o valoare întreagă reprezentînd codul de eroare depistat. Nu poate fi citată în modul direct.

De obicei e utilizată sub forma:

IF ERR = ⟨cod eroare⟩ **THEN** . . .

Cuvinte cheie asociate. **ERL, ERROR, ON ERROR**

ERROR(S)

ERROR ⟨coden⟩
10 I%=15
20 **ERROR** I%
30 **END**
RUN
String too long in line 20

Funcţie. Simulează apariţia unei erori GW–BASIC avînd codul ⟨coden⟩ (expresie numerică întreagă 0–255). Dacă nu eroarea este direct recunoscută de GW–BASIC, ea este tratată în conformitate cu opţiunile precizate într-un enunţ **ON ERROR** anterior, altminteri se tipăreşte mesajul 'Unprintable error'.

Cuvinte cheie asociate. **ERL, ERR, ON ERROR**

EXP(F)

EXP ⟨expresie numerică⟩
10 X=5
20 **PRINT EXP** (X–1)
RUN
54.59815

Funcţie. Calculează puterea dată de argumentul ⟨expresie numerică⟩ pentru baza logaritmilor naturali.

FIELD(S)

FIELD [#] ⟨numărfis⟩ ⟨listă cîmpuri⟩
cu
⟨listă cîmpuri⟩ : : = ⟨cîmp⟩ | ⟨listă cîmpuri⟩, ⟨cîmp⟩
⟨cîmp⟩ : : = ⟨lungime⟩ **AS** ⟨varsir⟩

10 **REM** DEFINEŞTE ÎN FIŞIERUL PERS ÎNREG DE 32 OCTETI
20 **OPEN** "R", #1, "A : PERS", 32
25 **REM** DEFINEŞTE ÎNREGISTRARE DE GARDĂ CU NUMĂR SALARIAŢI
30 **FIELD** #1, 2 **AS** NRSAL$, 30 **AS** FILLER$
35 **REM** DEFINEŞTE ÎNREGISTRARE SALARIAT – RETRIBUŢIE
40 **FIELD** #1, 25 **AS** NUME$, 7 **AS** RETRIB$
50 **GET** #1 'CITIRE ÎNREGISTRARE GARDA'
60 NRPERS % = **CVI** (NRSAL$)
70 **FOR** I%=1 **TO** NRPERS
80 **GET** #1, I%
90 **PRINT** NUME$, **CVS** (RETRIB$)
100 **NEXT** I%

Enunţ. Pentru un fişier aleator desemnat cu ⟨numărfis⟩ defineşte o înregistrare constituită din cîmpuri identificate cu variabila tip şir ⟨varsir⟩ din ⟨lista de cimpuri⟩, fiecare avînd un număr de caractere egal cu ⟨lungime⟩. În acelaşi timp variabilelor ⟨varşir⟩ li se alocă un spaţiu special. Lungimea totală nu trebuie să depăşească pe cea definită în enunţul **OPEN** pentru acelaşi ⟨numărfis⟩.

Cuvinte cheie asociate. **GET, LSET, PUT, RSET**

FILES(C)

FILES [⟨specificator fişier⟩]
REM AFIŞARE FIŞIERE DE PE UNITATEA A:
FILES "A : "
REM AFIŞARE FIŞIERE DE TIP BAS ÎN CATALOG CURENT
FILES "* · BAS"
REM AFIŞARE FIŞIER CATALOG SUBCAT 11 ⟨DIR⟩
FILES "\SUBCAT1\SUBCAT11"

Comandă. Afişarea numelui complet al unor fişiere din catalogul curent sau specificat în mod explicit, utilizînd expresiile şir ⟨specificator fişier⟩.

FIX(F)

FIX (⟨expresie numerică⟩)
PRINT FIX (67.512)
67

Funcţie. Calculează partea întreagă trunchiată a argumentului real ⟨expresie numerică⟩.

FOR . . . NEXT(S)

FOR ⟨var control⟩=⟨val in⟩ **TO** ⟨val fin⟩ [**STEP** ⟨increment⟩]
.
.
.
.
.
NEXT [⟨lista var control⟩]

```
10 REM ÎNMULŢIRE MATRICE CU VECTOR ŞI TIPĂRIRE
20 DEFINT A, X, Y, I, J
30 DIM A(20, 30), X(30), Y(20)
40 REM INIŢIALIZARE VECTOR X CU RANGUL I ŞI
45 REM MATRICE A CU PRODUS RANGURI I * J
50 FOR I=1 TO 30
60 X(I)=I
70 FOR J=1 TO 20
80 A(I,J)=I * J
90 NEXT J, I
100 REM PRODUS MATRICE VECTOR
110 FOR J=1 TO 20
120      Y(J)=0
130      FOR I=1 TO 30
140           Y(J)=Y(J)+A(I,J) * X(I)
150      NEXT I
160      PRINT Y(J)
170 NEXT J
180 END
```

Enunţ. Dacă variabila de control are o valoare ce nu depăşeşte valoarea expresiei ⟨valfin⟩ se execută întregul set de enunţuri ce urmează enunţului **FOR** pînă la primul enunţ **NEXT** corespondent, pornindu-se iniţial cu valoarea expresiei ⟨valin⟩. În caz contrar se trece direct la enunţul ce urmează enunţului **NEXT.** După execuţia acestui set de enunţuri se calcu-

lează valoarea ⟨var control⟩+⟨increment⟩ care devine noua valoare a variabilei de control, unde ⟨increment⟩ este o expresie avînd valoarea implicită 1 (putînd lua şi valori negative). Dacă valoarea expresiei ⟨valfin⟩ este situată între vechea valoare a variabilei de control şi cea nouă, atunci se execută enunţul următor enunţului **NEXT** corespondent, în caz contrar se reia execuţia setului de enunţuri începînd cu cel următor enunţului **FOR.** Buclele **FOR . . . NEXT** pot fi incluse una în cealaltă, dacă punctul de sfîrşit coincide, putîndu-se face comasarea enunţurilor NEXT, citînd variabilele de control ce identifică buclele în ordinea descrescătoare a nivelului de incluziune.

FRE(F)

```
FRE ⟨vid⟩
PRINT FRE (Ø)
  14542
```

Funcţie. Se determină numărul de octeţi liberi în memorie, deci neutilizaţi de către GW–BASIC. Argumentul ⟨vid⟩ este un argument fictiv (pentru păstrarea caracterului de funcţie) totuşi **FRE** (" ") forţează în prealabil o colecţie "gunoi".

GET(S) Fişiere de comunicaţii

```
GET [#] ⟨număr canal⟩, ⟨lungime⟩
10 REM DESCHIDE CANAL COMUNICAŢII 1, VITEZA 9600,
   1 BIT STOP, 8 BITI DATE ÎN MOD BINAR
20 OPEN "COM 1 : 9600, N, 1 BIN" AS # 2
30 GET #2, 80
```

Enunţ. Citeşte un număr de octeţi egal cu ⟨lungime⟩ în zona tampon de comunicaţii asociată la ⟨număr canal⟩. Valoarea ⟨lungime⟩ nu poate depăşi pe aceea declarată la enunţul **OPEN** pentru fişierul ⟨număr fişier⟩.

Cuvinte cheie asociate. **OPEN** (varianta **COM**)

GET(S) Fişiere aleatoare

```
GET [#] ⟨număr canal⟩ [,⟨număr înregistrare⟩]
 5 PRINT "NR. CRT", "NUME–PRENUME", "RETRIBUŢIE"
10 OPEN "R", 1, "PERS", 32
15 FIELD 1, 2 AS NRSAL$, 30 AS FILLER$
20 FIELD 1, 25 AS NUME$, 7 AS RETRIB$
25 GET 1
30 NR=CVI (NRSAL$)
35 FOR J=1 TO NR
40   GET 1, J
50   PRINT J, NRSAL$, RETRIB$
60 NEXT J
65 CLOSE 1
70 END
RUN
```

```
NR.CRT   NUME-PRENUME            RETRIBUȚIE
  1        GEORGESCU CARMEN       2786.35
  .
  .
  .
 124      VASILE OLIMPIU          4583.99
```

Enunț. Citește înregistrarea de rang ⟨număr înregistrare⟩ de pe fișierul disc în acces aleator, desemnat prin ⟨număr canal⟩ într-o zonă tampon descrisă prin intermediul unor enunțuri **FIELD** asociate. Dacă este omis ⟨număr înregistrare⟩ (expresie numerică întreagă 1–16.777.216) se citește înregistrarea următoare celei curente. Se poate afla conținutul înregistrării citite sau cu variabilele din enunțurile **FIELD** asociate, ori cu enunțuri **INPUT#** ori **INPUT LINE#**. Se recomandă și testarea condiției de sfîrșit de fișier cu funcția logică **EOF**.

Cuvinte cheie asociate: **EOF, FIELD, INPUT, INPUT LINE, OPEN, PUT**

GET (SGR) Imagini grafice

GET [**STEP**] (⟨x1⟩, ⟨y1⟩) – [**STEP**] (⟨x2⟩, ⟨y2⟩), ⟨masiv⟩

```
290 DIM M%(20)
300 GET (31, 74) - (41, 62), M%
```

Enunț. Se transferă imaginea grafică delimitată de un dreptunghi cu laturi paralele cu marginile ecranului, avînd coordonatele NE date de expresiile ⟨x1⟩ pentru abscisă și ⟨y1⟩ pentru ordonată și SV date de expresiile ⟨x2⟩ pentru abscisă și ⟨y2⟩ pentru ordonată, într-un ⟨masiv⟩ de tip întreg. Dimensiunea masivului în octeți este

4+INT ((x * bpp+7)/8) * y

unde:

bpp (bits per pixel)=
1 (rezoluție mare și foarte mare) sau
2 (rezoluție medie)

x – dimensiunea (în puncte) dreptunghiului pe orizontală
y – dimensiunea (în puncte) dreptunghiului pe verticală

Fiecare rînd de elemente de imagine este aliniat stînga la limită de octet.

Cuvinte cheie asociate. **PUT**

GOSUB ... RETURN (S)

GOSUB ⟨număr linie 1⟩
.
.
.
RETURN ⟨număr linie 2⟩

```
10 GOSUB 40
20 PRINT "REVENIRE DIN SUBRUTINĂ"
30 END
40 PRINT "INTRARE ÎN SUBRUTINĂ"
50 RETURN
```

RUN
INTRARE ÎN SUBRUTINĂ
REVENIRE DIN SUBRUTINĂ

Enunţ. **GOSUB** transferă controlul la linia ⟨număr linie 1⟩ de unde se presupune că începe o subrutină GW–BASIC.

RETURN, care poate logic apare oriunde în subrutină, produce ieşirea din subrutină şi revenirea la linia ⟨număr linie 2⟩, sau în lipsa acesteia la linia imediat următoare enunţului **GOSUB.** Dacă este prezent, ⟨număr linie 2⟩, trebuie să difere de orice linie a subrutinei. În momentul apariţiei enunţului **RETURN** nu trebuie să existe enunţuri **FOR, GOSUB, WHILE** nerezolvate.

GOTO(S)

GOTO ⟨număr linie⟩
.
.
.
.
20 I%=0
30 **READ** R
40 **PRINT** "RAZA="; R
50 A=3.14 * R ∧ 2
60 **PRINT** "ARIA="; A
70 I%=I%+1
80 **IF** (I%=<3) **THEN GOTO** 30
90 **DATA** 5, 7, 12
RUN

RAZA=5	ARIA= 78.5
RAZA=7	ARIA=153.86
RAZA=12	ARIA=452.16

Enunţ. Controlul se transferă la primul enunţ executabil de pe o linie egală ori mai mare ca ⟨număr linie⟩.

GWBASIC(DC)

GWBASIC [⟨specificator fişier⟩] [⟨stdin⟩] [[⟨stdout⟩]
[/F : ⟨număr canal⟩ [/S : ⟨maxbuf⟩] [/C : ⟨bufsiz⟩]
[/M : ⟨high⟩] [,⟨blocsiz⟩]] [/D] [/I]]

Utilizare: 4 fişiere şi 32 K memorie.

A>GWBASIC/F: 4/M: 32 768. Datele citite cu **INPUT** şi **LINE INPUT** provin din fişierul **FISIN** iar datele imprimate cu **PRINT** se scriu în **FISOUT.**

Comanda DOS. Iniţializează mediul de operare GW–BASIC. Opţional se va executa programul BASIC ⟨specificator fişier⟩ (expresie tip şir neinclusă în ghilimele), cu eventuala redirectare a fişierului standard de intrare pe ⟨stdin⟩ (expresie tip şir neinclusă în ghilimele) ori a fişierului standard de ieşire pe ⟨stdout⟩ (expresie de tip şir neinclusă în ghilimele). Se poate fixa o limită ⟨număr canal⟩ pentru numărul de fişiere simultan deschise. (Fiecare fişier cere 62 octeţi pentru blocul control fişier FCB, la care se adaugă valoarea ⟨maxbuf⟩, implicit 128). Valoarea sa implicită este 3. (Unii o recomandă 6, dar depinde şi de opţiunea FILES dun CONFIG. SIS activat

la lansarea MS—DOS. Evident, valoarea ⟨maxbuf⟩ (maximum 32.767 octeţi) nu poate fi depăşită de valorile citate în **OPEN.**

Opţiunea /D indică cerinţa de rezidenţă a pachetului de rutine matematice pentru funcţii transcendente (ATN, COS, EXP, LOG, SIN, SQR, TAN) în dublă precizie, cu riscul unui plus de 3 000 octeţi. Opţiunea /I permite alocarea statică a spaţiului necesar pentru operaţii la nivel de fişier. Numai atunci opţiunile /F şi /S au efect. Totodată GW—BASIC va aloca pentru segmentul de date şi stivă ⟨high⟩ octeţi (implicit 64 K). Dacă se intenţionează încărcarea (via comanda SHELL) de programe peste spaţiul de lucru al lui GW—BASIC trebuie rezervat spaţiu de lucru pentru programe şi date la nivelul a ⟨blocsiz⟩ paragrafe (implicit &H1000).

Argumentul ⟨bufsiz⟩ are sens numai dacă este prezentă facilitatea de comunicaţii RS232 pentru a preciza zona tampon de recepţie cartelă (max 32 767, implicit 256).

Cuvinte cheie asociate. **INPUT, LINE INPUT, PRINT, SHELL**

HEX$ (F)

HEX$ (⟨expresie numerică⟩

```
10 INPUT I%
20 PRINT I% "ZECIMAL ESTE" HEX$ (I%) "HEXAZECIMAL"
RUN
   42
42 ZECIMAL ESTE 2A HEXAZECIMAL
```

Funcţie. Se returnează un şir care reprezintă valoarea hexazecimală a argumentului ⟨expresie numerică⟩. În prealabil are loc rotunjirea expresiei şi în caz că valoarea e negativă, are loc determinarea formei complementului faţă de 2.

Cuvinte cheie asociate. **OCT$**

IF ... GOTO ... ELSE (S)

IF ⟨condiţie⟩ **GOTO** ⟨număr linie 1⟩
[**ELSE** ⟨variantă⟩]

cu

⟨variantă⟩: : =⟨listă enunţuri⟩ | ⟨număr linie⟩
⟨listă enunţuri⟩: : =⟨enunţ⟩ | ⟨listă enunţuri⟩ : ⟨enunţ⟩

```
10 REM CALCULUL SUMEI A 5 NUMERE
20 I%=1
25 S%=0
30 INPUT N%
40 S%=S%+N%
45 I%=I%+1
50 IF(I%=<5) GOTO 30 ELSE PRINT S%
```

Enunţ. Se evaluează condiţia ⟨condiţie⟩. Dacă este adevărată, controlul este transferat primului enunţ executabil începînd cu linia ⟨număr linie 1⟩. În caz contrar, dacă opţiunea **ELSE** este prezentă se execută enunţurile din ⟨lista enunţuri⟩ sau se transferă controlul primului enunţ executabil începînd cu linia ⟨număr linie 2⟩. Altminteri (condiţie neadevărată şi

opţiunea **ELSE** absentă) nu se execută nici un enunţ. Desigur numai ultimul enunţ din ⟨lista enunţuri⟩ poate fi de tip **GOTO**.

Cuvinte cheie asociate. **IF . . . THEN . . . ELSE**

IF . . . THEN . . . ELSE(S)

IF ⟨condiţie⟩ **THEN** ⟨varianta 1⟩ **ELSE** ⟨varianta 2⟩
cu:
⟨varianta 1⟩: : =⟨lista enunţuri 1⟩ | ⟨număr linie 1⟩
⟨varianta 2⟩: : =⟨lista enunţuri 2⟩ | ⟨număr linie 2⟩
⟨lista enunţuri⟩: : =⟨enunţ⟩ | ⟨lista enunţuri⟩: ⟨enunţ⟩

```
10 REM CALCUL SUMA 5 VALORI ABSOLUTE A 5 NUMERE
15 S%=0
20 FOR I=1 TO 5
30    INPUT N%
40    IF (N%>=0) THEN S%=S%+N% ELSE S%=S%-N%
50 NEXT I
60 PRINT S%
```

Enunţ. Se evaluează condiţie ⟨condiţie⟩. Dacă este adevărată, se execută enunţurile din ⟨lista enunţuri 1⟩, sau se transferă controlul primului enunţ executabil începînd cu linia ⟨număr linie 1⟩. Dacă condiţia nu este adevărată şi opţiunea **ELSE** este prezentă, se execută enunţurile din ⟨lista enunţuri 2⟩ sau se transferă controlul primului enunţ executabil începînd cu linia ⟨număr linie 2⟩. Altminteri (condiţie falsă şi opţiunea **ELSE** absentă), nu se execută nici un enunţ.

Cuvinte cheie asociate. **IF . . . GOTO . . . ELSE**

INKEY$(F)

INKEY$

```
  5 REM SE TASTEAZĂ ÎN MOD NESOLICITAT CARACTERE
 10 SIR$=INKEY$
 15 REM DACĂ TAMPONUL NU CONŢINE CARACTER
    SE REIA DE LA 10
 20 REM DACĂ E UN CARACTER NOU ASCII, DECI O CHEIE
    FUNCŢIONALĂ SE TRECE LA LINIA 100
 30 IF LEN (SIR$) GOTO 10 ELSE IF LEN (SIR$)=2 GOTO 100
 40 PRINT HEX$ (ASC (SIR$))
 50 GOTO 10
100 SIRA$=MID$ (SIR$, 1, 1)
110 SIRB$=MID$ (SIR$, 2, 1)
120 PRINT HEX$ (ASC (SIRA$)); " "; HEX$ (ASC (SIRB$)
```

Funcţie. Dacă a apărut o intrare nesolicitată de la consolă sub forma unui caracter ASCII standard, se restituie utilizatorului un şir de 1 octet cuprinzînd acest caracter. Dacă s-a tastat o cheie funcţională, sau o combinaţie de chei, se restituie un şir de două caractere, primul fiind **NULL**, celălalt codul extins. Dacă nu s-a tastat nici un caracter se returnează un şir conţinînd caracterul **NULL**.

Cuvinte cheie asociate. **KEY** n

INP(F)

INP ⟨port⟩
10 I%=**INP** (36763)

Funcţie. Se determină un octet citit de la poarta ⟨port⟩ (expresie numerică întreagă pozitivă: 0÷65535).

Cuvinte cheie asociate. **OUT**

INPUT(S)

INPUT [;] [⟨prompt⟩;] ⟨lista de variabile⟩
10 **INPUT** "TASTAŢI VALOARE UNGHI"; U
20 **PRINT SIN** (U)
RUN
  TASTAŢI VALOARE UNGHI ? 1.570795
  1.00000

Enunţ. La consolă se afişează caracterul '?' (semnul întrebării), opţional precedat de constanta şir ⟨prompt⟩, aşteptîndu-se introducerea de date, separate de virgule, care se atribuie variabilelor din ⟨lista de variabile⟩, cu condiţia respectării tipului acestor variabile şi concordanţei acestuia cu lungimile implicite.

– Dacă după ⟨prompt⟩ separatorul este virgula, nu se mai afişează caracterul '?'.
– Dacă după cuvîntul cheie **INPUT** apare separatorul ';' (punct şi virgulă), atunci acţionarea terminatorului ⟨CR⟩ nu conduce în ecou la secvenţa **⟨CR⟩ ⟨LF⟩**.

Cuvinte cheie asociate. **INPUT#, INPUT$, LINE INPUT**

INPUT#(S)

INPUT# ⟨număr canal⟩, ⟨lista de variabile⟩
10 **OPEN** "FIS TEXT" **FOR INPUT AS** #1
20 **INPUT** #1, X$, Y$, Z$

Enunţ. Datele sînt citite din fişierul secvenţial ⟨număr canal⟩ (desemnat printr-un enunţ **OPEN** anterior) în lista de variabile ⟨lista de variabile⟩. Fişierul ⟨număr canal⟩ trebuie să conţină datele ca şi cum s-ar fi răspuns la enunţul **INPUT.** Se remarcă situaţiile:

a) Pentru date numerice: se ignoră caracterele iniţiale tip spaţiu, **⟨CR⟩**, **⟨LF⟩**, terminatorul fiind reprezentat de spaţiu, **⟨CR⟩**, **⟨LF⟩**, virgulă.

b) Pentru date şir: se ignoră caracterele iniţiale tip spaţiu, **⟨CR⟩**, **⟨LF⟩**. Sînt două situaţii:
– constanta şir este inclusă între semnele de citare "⟨constantă şir⟩"
– constanta nu e inclusă între semnele citării, ⟨constantă şir⟩ ⟨terminator⟩,
unde:

⟨terminator⟩ : : = **⟨CR⟩** | **⟨LF⟩** | ⟨virgula⟩

Cuvinte cheie asociate. **INPUT, INPUT$, LINE INPUT, OPEN**

INPUT$(F)

INPUT$ (⟨lungime⟩ [,[#] ⟨număr canal⟩])
10 **REM** TIPĂRIRE CONŢINUT FIŞIER SECVENŢIAL ÎN OCTAL
20 **OPEN** "I", 1, "FIŞIER"

```
30 IF EOF (1) THEN 60
40 PRINT OCT (ASC (INPUT$ (1, #1)));
50 GOTO 30
60 END
```

Funcţie. De la dispozitivul standard de intrare sau de la consolă, ori din fişierul ⟨număr canal⟩ se citeşte un număr de caractere ⟨lungime⟩ (expresie întreagă), oricare ar fi acestea (mai puţin **CTRL/BREAK** care întrerupe execuţia funcţiei). Dacă intrarea se face de la tastatură, nu apare ecoul.

Cuvinte cheie asociate. **INPUT, INPUT#, LINE INPUT, OPEN**

INSTR(F)

INSTR ([⟨start⟩,] ⟨şir⟩ ⟨subşir⟩)

```
10 X$="UN EXEMPLU DE SUBŞIR ÎN ŞIR"
20 Y$="EXEMPLU"
30 PRINT INSTR (2, X$, Y$)
RUN
4
```

Funcţie. Şirul de caractere ce reprezintă valoarea expresiei ⟨subşir⟩ este căutat pentru coincidenţă în cuprinsul şirului reprezentînd valoarea expresiei ⟨şir⟩, pornind cu poziţia ⟨start⟩, (expresie întreagă, 1–255, implicit 1). Poziţia în care se face identificarea constituie valoarea funcţiei. În particular ea va fi 0 dacă punctul de ⟨start⟩ e mai mare ca lungimea valorii expresiei ⟨şir⟩, sau ⟨şir⟩ are o valoare vidă, ori ⟨subşir⟩ nu aparţine expresiei ⟨şir⟩. De asemenea un ⟨subşir⟩ vid este întotdeauna identificat în poziţia ⟨start⟩ (implicit 1).

INT (F)

INT (⟨expresie numerică⟩)

```
PRINT INT (-15.07); INT (35.24)
-16  35
```

Funcţie. Calculează cea mai mare valoare întreagă care nu depăşeşte valoarea expresiei ⟨expresie numerică⟩.

Cuvinte cheie asociate. **CINT, FIX.**

IOCTL(S)

IOCTL [#] ⟨număr canal⟩, ⟨şir⟩

```
10 REM TRATEAZĂ DRIVER DE IMPRIMANTĂ CU LUNGIME PAGINĂ
15 REM DE 50 LINII
20 OPEN "\DEV\LPT 1" FOR OUTPUT AS#2
30 IOCTL #2, "PL50"
```

Enunţ. Expresia ⟨şir⟩ (max. 255 caractere) conţinînd date de control pentru un modul de control periferic (driver), anterior deschis ca fişierul ⟨număr canal⟩ este trimisă către acesta.

Observaţie. Nu toate modulele de control periferice pot primi comanda **IOCTL.**

Cuvinte cheie asociate. **IOCTL$, OPEN.**

IOCTL$(F)

```
IOCTL$ ([#]⟨număr canal⟩)
10 REM TRATEAZĂ DRIVER DE IMPRIMANTĂ CU LUNGIME
   PAGINĂ DE 50 LINII
20 OPEN "\DEV\LPT1" AS OUTPUT AS #2
30 PRINT IOCTL$(#2)
   LP50
```

Funcție. Restituie utilizatorului un șir de 'date control' privind un modul control periferic (driver) relativ la dispozitivul de tip caracter asociat la deschidere numărului ⟨număr canal⟩.

Observație. Nu toate modulele de control periferic prelucrează și recunosc șiruri de control.

Cuvinte cheie asociate. **IOCTL, OPEN**.

KEY LIST(S)

```
KEY LIST
KEY LIST
F1 – LIST     F2 – RUN ⟨CR⟩...
```

Enunț. Afișează toate valorile cheilor soft, utilizînd 15 caractere pentru fiecare cheie în parte.

Cuvinte cheie asociate. **KEY OFF, KEY ON, KEY** n

KEY OFF(S)

```
KEY OFF
10 KEY OFF
20 LOCATE 25, 7 "POZIȚIONARE CURSOR ÎN LINIA 25, COLOANA 7"
30 PRINT "S-A ANULAT AFIȘAREA CHEILOR SOFT"
```

Enunț. Șterge de pe ultima linie a ecranului afișajul tuturor cheilor software, dînd posibilitate utilizatorului să folosească această linie pentru a scrie date pe ea.

Cuvinte cheie asociate. **KEY LIST, KEY ON, KEY** n

KEY ON(F)

```
KEY ON
KEY ON
```

Enunț. Primele 6 caractere ale valorilor tuturor celor 10 chei sînt afișate pe ultima linie a ecranului.

Cuvinte cheie asociate. **KEY LIST, KEY OFF, KEY** n

KEYn(S)

```
KEY ⟨n⟩, ⟨expresie șir⟩
10 REM STABILIRE VALORI CHEI SOFT 1–5 CU NUME SIMBOLURI
   ORGANIGRAMA
20 KEY OFF 'ANULARE AFIȘARE CHEI SOFT
30 DATA "PĂTRAT", "DREPT", "ROMB", "CERC", "SĂGEAT", "LINIE"
40 FOR I%=1 TO 5
50    READ SOFTK$ (I%)
```

60 **KEY** I%, SOFTK$ (I%)
70 **NEXT** I%
80 **KEY ON** 'RESTABILIRE AFIŞARE CHEI SOFT

Enunţ. Cheii soft de rang ⟨n⟩ (expresie numerică cu valoare 1–10) i se atribuie primele 15 caractere din valoarea expresiei şir ⟨expresie şir⟩. Această valoare va fi acceptată la acţionarea cheii funcţionale F ⟨n⟩ şi succesiv cîte un caracter din acest şir va fi restituit la fiecare invocare a funcţiei **INKEY$**.

Cuvinte cheie asociate. **KEY LIST, KEY OFF, KEY ON**

KEY

KEY ⟨n⟩, **CHR$** (⟨shift⟩)+**CHR$** (⟨cod baleiaj⟩)
cu:

a) ⟨shift⟩ este un cod avînd valorile:

CAPSLOCK	– &H40
NUMLOCK	– &H20
CTRL	– &H04
SHIFT (right)	– &H01
SHIFT (left)	– &H02

b) ⟨cod baleiaj⟩ – valoare zecimală 1–83 (tastatura tip 1) sau 1–103 (tastatura tip 2).

Enunţ. Cheii utilizator ⟨n⟩ (expresie întreagă cu valori 15–20) i se atribuie valoarea expresiei şir de după virgulă. În consecinţă va apare o întrerupere specifică la acţionarea oricărei asemenea chei.

Cuvinte cheie asociate. **KEY LIST, KEY OFF, KEYn OFF, KEY ON, KEYn ON, KEYn STOP**

KEYn OFF(S)

KEY (⟨n⟩) **OFF**

Enunţ. Întreruperea provocată de acţionarea cheii soft ⟨n⟩ (expresie numerică întreagă cu valori 1–10), ori a cheii utilizator ⟨n⟩ (expresie întreagă cu valori 15–20), sau a cheii de acţionare cursor ⟨n⟩ (expresie întreagă cu valori 11–14) este dezactivată (inhibată).

Cuvinte cheie asociate. **KEYn, KEYn ON, KEYn STOP, ON KEY**

KEYn ON(S)

KEY (⟨n⟩) **ON**

10 **KEY** 4, **SCREEN** 0, 0, 0, 'STABILIRE VALOARE CHEIE SOFT 4
20 **KEY** (4) **ON** 'ACTIVARE ÎNTRERUPERE ACŢIONARE CHEIE 4
:
:
150 **ON KEY** (4) **GOSUB** 1100 'STABILIRE RUTINĂ UTILIZATOR TRATARE ÎNTRERUPERE

Acţionare cheie 4
:
:
1100 **REM** RUTINĂ TRATARE ÎNTRERUPERE

Enunț. Se activează recunoașterea întreruperii provocată de acționarea cheii soft ⟨n⟩ (expresie numerică întreagă cu valori 1–10), ori a cheii utilizator ⟨n⟩ (expresie întreagă cu valori 15–20), sau a cheii de acționare cursor ⟨n⟩ (expresie întreagă cu valori 11–14).

Cuvinte cheie asociate. **KEYn, KEYn OFF, KEYn STOP, ON KEY**

KEYn STOP(S)

KEY [⟨n⟩] **STOP**

Enunț. Se suspendă (pînă la reactivare cu **KEY** (⟨număr⟩) **ON**) tratarea întreruperii provocate de acționarea cheii soft ⟨n⟩ (expresie numerică cu valori 1–10), ori a cheii utilizator ⟨n⟩ (expresie numerică cu valori 15–20), sau a cheii de acționare a cursorului ⟨n⟩ (expresie numerică cu valori 11–14).

Cuvinte cheie asociate. **KEYn, KEYn OFF, KEYn STOP, ON KEY**.

KILL (C)

KILL ⟨fișier⟩

```
 5 KILL "STOC. DAT" 'ȘTERGE FIȘIERUL STOC. DAT DIN CATALOGUL
   CURENT, DISCUL IMPLICIT
10 REM ȘTERGE FIS. DAT DE PE CALEA CATAL 1-CATAL 2 A CATALOGU-
   LUI CURENT, DISCUL A
15 KILL "A : CATAL 1\CATAL 2\FIS. DAT"
```

Comanda. Șterge fișierul de pe disc ⟨fișier⟩ (expresie șir, desemnînd un specificator de fișier complet, ori cu componente implicite), cu condiția ca acesta să nu fie deschis.

LCOPY(C)

LCOPY [⟨număr⟩]

Comanda. Descarcă conținutul ecranului pe imprimanta implicită. Expresia numerică ⟨număr⟩ nu are semnificație în GW-BASIC fiind utilizată pentru compatibilitate cu alte versiuni de interpretor BASIC.

LEFT$(F)

LEFT$ (⟨expresie șir⟩, ⟨lungime⟩)

```
10 X$="EXEMPLU"
15 FOR I=1 TO 6
20    PRINT LEFT$ (X$, I-1)
25 NEXT I
RUN
   Null
   E
   EX
   EXE
   EXEM
   EXEMP
```

Funcție. Furnizează un subșir prefix de lungime ⟨lungime⟩ (expresie întreagă cu valori 0–255) al șirului ⟨expresie șir⟩. Dacă valoarea ⟨lungime⟩ este mai mare ca lungimea efectivă a șirului ⟨expresie șir⟩ atunci întregul șir inițial este restituit. Semnificația valorii 0 a expresiei ⟨lungime⟩ este șirul **NULL**.

Cuvinte cheie asociate. **MID$, RIGHT$**

LEN (F)

LEN (⟨expr şir⟩)
X$="LUNGIMEA ACESTUI ŞIR ESTE : "
PRINT X$; **LEN** (X$)
LUNGIMEA ACESTUI ŞIR ESTE : 28

Funcţie. Furnizează lungimea valorii expresiei şir ⟨expr şir⟩⟩ Dacă şirul este **NULL** valoarea restituită este zero.

LET (S)

[**LET**]⟨variabilă⟩=⟨expresie⟩
10 **LET** J%=17
20 **LET** X$="EXEMPLU"
30 **LET** J%=**LEN** (X$)
40 **LET** W=(A+B)*S∧2+E+12.5

Enunţ. Atribuie valoarea expresiei ⟨expresie⟩ variabilei ⟨variabila⟩ – cu respectarea concordanţei de tip.

LINE (SGR)

LINE [[**STEP**](⟨x1⟩, ⟨y1⟩)]–[**STEP**](⟨x2⟩, ⟨y2⟩)[, [⟨culoare⟩][, **B**[**F**]][, ⟨stil⟩]]
10 **LINE** (10, 10)–(200, 200) 'TRASEAZĂ DIAGONALĂ (10, 10)–(200, 200)
20 **LINE** – (200, 40), 3 'TRASEAZĂ ÎN CONTINUARE VERTICALA (200, 200)–(200, 40) CU CULOAREA 3
25 **REM** DREPTUNGHIUL DE COLŢURI SV (50, 60) ŞI NE (100, 120) E UMPLUT CU CULOAREA 2
30 **LINE** (50, 60)–(100, 120) 2, **BF**
35 **LINE** (5, 5) – **STEP** (–2, 45) 'TRASARE LINIE DIN (5, 5) ÎN (3, 50)

Enunţ. În lipsa opţiunilor **B** ('box' – dreptunghi) şi **F** ('Fill' – umplere) are loc trasarea unei linii din coordonatele absolute (sau relative faţă de punctul grafic curent dacă **STEP** e prezent) de abscisă ⟨x1⟩ şi ordonată ⟨y1⟩ (expresii numerice întregi) în punctul de abscisă ⟨x2⟩ şi ⟨y2⟩ (expresii numerice întregi) şi stilul ⟨stil⟩ (masă întreagă de 16 biţi, reprezentînd şablonul de aprindere a punctelor).

Dacă enunţul începe cu separatorul ' – ' se presupune că trasarea se face din punctul curent.

Opţiunea **B** (box) semnifică trasarea dreptunghiului de coordonate (absolute ori relative) (x1, y1), (x2, y1), (x2, y2), (x1, y2) cu culoarea ⟨culoare⟩ şi stilul ⟨stil⟩, iar opţiunea **BF** înseamnă umplerea susamintitului dreptunghi cu culoarea ⟨culoare⟩ (în acest din urmă caz parametrul ⟨stil⟩ nu are nici o semnificaţie).

Cuvinte cheie asociate. **CIRCLE, COLOR**

LINE INPUT (S)

LINE INPUT [;][⟨prompt⟩;]⟨var şir⟩
10 **REM** CREARE FIŞIER AGENDĂ
20 **OPEN** "O", 1, "FISNUM"
25 **PRINT** #1, 20
30 **FOR** I=1 **TO** 20
40 **LINE INPUT** "NUME ŞI PRENUME :"; NUMP$

```
50 PRINT #1, NUMP$
60 NEXT I
70 CLOSE #1
```

Enunț. Variabila ⟨var șir⟩ recepționează un șir de max 254 caractere introdus de la tastatură, eventual ca urmare a afișării unui mesaj ⟨prompt⟩. Execuția enunțului poate fi anulată prin tastarea **CTRL/BREAK.**

Dacă separatorul ';' urmează nemijlocit cuvîntului cheie **INPUT,** terminatorul de linie ⟨**CR**⟩ nu va avea ca ecou pe ecran secvența ⟨**CR**⟩ ⟨**LF**⟩. Același caracter nu va avea rol de terminator în secvența ⟨**LF**⟩ ⟨**CR**⟩, în schimb numai caracterul ⟨**LF**⟩ va fi transmis variabilei receptoare.

Cuvinte cheie asociate. **INPUT, INPUT$, INPUT#, LINE INPUT#**

LINE INPUT#(S)

LINE INPUT# ⟨număr canal⟩, ⟨var șir⟩

```
10 REM TIPĂRIRE FIȘIER AGENDĂ
20 OPEN "I",1, "FISNUM"
30 LINE INPUT #1, NP$ 'NUMĂR ÎNREGISTRĂRI
40 FOR I%=1 TO CVI (NP$)
50    LINE INPUT #1, NUMP$
60    PRINT I%; NUMP$
70 NEXT I%
80 CLOSE #1
```

Enunț. Se citește de la fișierul secvențial ⟨număr canal⟩ un șir de caractere terminat cu caracterul ⟨**CR**⟩ (exceptînd secvența ⟨**CR**⟩ ⟨**LF**⟩ care este ignorată și secvența ⟨**LF**⟩ ⟨**CR**⟩ care este păstrată ca atare) și care se atribuie variabilei șir ⟨var șir⟩.

Cuvinte cheie asociate. **INPUT$, INPUT#, LINE INPUT, INPUT**

LIST(C)

LIST [⟨număr linie 1⟩] [–[⟨număr linie 2⟩]] [,⟨specificator fișier⟩]

LIST, LPT1: 'LISTARE PROGRAM LA IMPRIMANTĂ

LIST 10–80 'LISTAREA PE ECRAN A LINIILOR 10–90

Comandă. Se listează în cadrul fișierului specificat cu ⟨specificator fișier⟩ (implicit ecranul, adică **SCRN:**) programul sursă GW–BASIC curent, între liniile de numere (implicit prima linie) ⟨număr linie 1⟩ și ⟨număr linie 2⟩ (implicit ultima linie), unde simbolurile metalingvistice sînt expresii numerice întregi. În cazul în care listarea se face pe ecran sau imprimantă, aceasta se poate stopa tastînd **CTRL/BREAK.**

Cuvinte cheie asociate. **LLIST**

LLIST(C)

LIST [⟨număr linie 1⟩] [–[⟨număr linie 2⟩]]

LLIST 'LISTARE ÎNTREG PROGRAM

LLIST – 200 'LISTAREA LINIILOR DE LA ÎNCEPUT PÎNĂ LA LINIA 200

Comandă. Se listează la imprimantă pe 132 caractere per linie programul sursă curent, între liniile de numere ⟨număr linie 1⟩ (implicit prima linie) și ⟨număr linie 2⟩ (implicit ultima linie), unde simbolurile metalingvistice sînt expresii numerice întregi.

LOAD(C)

LOAD ⟨specificator fişier⟩ [,**R**]

LOAD "A: PROGUT" 'ÎNCARCĂ PROGRAMUL PROGUT DIN CATALOGUL CURENT DE PE DISCUL A

Comandă. Încarcă în memorie noul program GW–BASIC care rezidă pe disc prin specificatorul complet ⟨specificator fişier⟩, după ce în prealabil au fost şterse toate variabilele şi liniile programului curent şi au fost închise fişierele active ale acestuia, exceptînd cazul opţiunii **R** cînd fişierele sînt păstrate deschise şi are loc în acelaşi timp şi executarea programului.

Cuvinte cheie asociate: **RUN**

LOC(F)

LOC (⟨număr canal⟩)

100 **IF LOC** (2) > 100 **THEN STOP**

Funcţie. Furnizează poziţia curentă în fişierul deschis sub identificatorul ⟨număr canal⟩. Această poziţie curentă este:

a) fişiere secvenţiale: cîtul împărţirii la 128 al poziţiei curente în octeţi

b) fişiere în acces aleator: numărul înregistrării tocmai citite sau scrise

c) fişiere de comunicare: min (255, număr de caractere în coada de aşteptare); în plus fişierele de comunicaţii deschise în mod ASCII refuză plasarea în coadă a caracterului "sfîrşit de fişier".

LOCATE(SGR)

LOCATE [⟨rînd⟩] [,[⟨coloana⟩] [,⟨perioadă⟩] [,⟨formă⟩]

cu:

⟨forma⟩ : : = ⟨dimens⟩ | ⟨şablon⟩

⟨dimens⟩ : : = ⟨linie start⟩, ⟨linie end⟩

⟨şablon⟩ : : = ⟨linie şablon⟩, ⟨grilă⟩

10 **REM** CURSOR ÎN LINIA 5, COLOANA 20, ÎNĂLŢIME 7–4+1

20 **SCREEN** 2 'MOD REZOLUŢIE MARE

30 **LOCATE** 5, 20,, 4, 7 'CURSOR BLOC CU FIECARE LINIE ÎN ŞABLON OFFH

40 **LOCATE** , , , 105, &H82 'LINIA 5 CURSOR UTILIZATOR ARE ŞABLON 82H

Enunţ. Se stabilesc selectiv o serie de caracteristici opţionale pentru cursor:

a) *caracteristici poziţionale.* Cursorul se poate muta în poziţie diferită de ⟨rînd⟩ (expresie numerică cu valori 1–25) şi ⟨coloană⟩ (expresie numerică cu valori 1–40, sau 1–80)

b) *caracteristici de vizualizare.* Cursorul apare potrivit valorii expresiei numerice ⟨perioada⟩:

0 – afişarea sa inhibată (exceptînd modul direct şi enunţurile **INPUT**)

1 – afişare permanentă

2–10 – licărire cu durate de apariţie de ⟨perioada⟩/18, 75 secunde.

c) *caracteristici de dimensiune.* Matricea cursorului apare între liniile ⟨linie start⟩ şi ⟨linie end⟩ care sînt expresii numerice cu valori 0–15 pentru cursor utilizator şi de scriere forţată ori 16–31 (modulo 16) pentru cursor utilizator în exclusivitate.

d) *caracteristici de formă.* Se programează pentru matricea cursorului o anume ⟨linie şablon⟩ (expresie numerică avînd incrementată valoarea efectivă a liniei cu 50 pentru cursorul de serie forţată şi 100 pentru cursorul utilizator), în conformitate cu ⟨grila⟩ (expresie numerică reprezentînd starea poziţiilor binare care vor fi puse în operaţie **XOR** cu fondul ecranului).

Matricea cursorului este de 16×16 pentru rezoluţie medie, 8×8 pentru rezoluţie mare şi 16×8 pentru rezoluţie super. Deşi iniţial cursorul utilizator e inhibat, matricea şablonului său este încărcată cu bytes OFFH.

Cuvinte cheie asociate. **LINE, PSET, SCREEN**

LOCATE(STX)

LOCATE [⟨rînd⟩] [,[⟨coloană⟩] [,⟨cursor⟩ [,[⟨linie start⟩] [,⟨linie stop⟩]]]]
10 **REM** CURSOR UTILIZATOR ÎN LINIA 5, COLOANA 20, ÎNĂLŢIME 7–4+1
20 **LOCATE** 5, 20, 1, 36, 39

Enunţ. Se stabilesc selectiv o serie de caracteristici opţionale pentru cursor:

a) *caracteristici poziţionale.* Cursorul se poate muta în poziţia definită de ⟨rînd⟩ (expresie numerică cu valori 1–25) şi ⟨coloana⟩ (expresie numerică cu valori 1–40, ori 1–80).

b) *caracteristici de vizibilitate.* Cursorul apare potrivit valorii booleane ⟨cursor⟩ ca invizibil (valoare 0) sau vizibil (valoare nenulă).

c) *caracteristici dimensionale.* Matricile cursoarelor utilizator şi de scriere forţată apar între liniile ⟨linie start⟩ şi ⟨linie end⟩ care sînt expresii numerice avînd valoarea 0–13 pentru cursor utilizator şi de scriere forţată şi 32–45 (modulo 32) pentru cursor utilizator în exclusivitate.

– Înălţimea cursorului e de maximum 8 linii pentru monitoare color şi 14 pentru alb-negru.

– În lipsa argumentului ⟨linie stop⟩ se presupune că înălţimea este de o linie.

– Dimensiunile implicite sînt:

	color	alb-negru
• cursor scriere forţată color	6–7	12–13
• cursor inserare	4–7	7–13
• cursor utilizator	0–7	0–13

Cuvinte cheie asociate. **INKEY$, INPUT$**

LOF(F)

LOF (⟨număr canal⟩)
10 **OPEN** "FISPERS" **A$** #3
20 **PRINT** "FIŞIERUL ARE : " ; **LOF** (3); "OCTEŢI"

Funcţie. Calculează lungimea în octeţi a fişierului căruia i s-a asociat la deschidere numărul ⟨numărul canal⟩. În cazul fişierelor de comunicaţie expresia ⟨buf siz⟩ – **LOC** (⟨număr canal⟩) indică spaţiul liber în zona tampon de intrare de ⟨bufsiz⟩ octeţi (implicit 256).

Cuvinte cheie asociate. **LOC, OPEN**

LOG[(F)]

LOG (⟨expresie numerică⟩)
X=(6 * 5+10)/7
PRINT LOG (X)
1.860752

Funcţie. Calculează logaritmul natural al argumentului pentru ⟨expresie numerică⟩.

Cuvinte cheie asociate. **EXP**

LPOS[(F)]

LPOS (⟨printer⟩)
10 **REM** DACĂ S-A AJUNS ÎN COLOANA 70 SE IMPRIMĂ CR.
20 **IF LPOS** (1) > =71 **LPRINT CHR$** (13)

Funcţie. Determină poziţia curentă în zona tampon a imprimantei **LPT** ⟨printer⟩, unde ⟨printer⟩ este o expresie întreagă, avînd valorile 1–3.

LPRINT[(S)]

LPRINT [⟨listă de expresii⟩] [⟨terminator⟩]
cu:
⟨listă de expresii⟩ : : = ⟨expresie⟩ | ⟨listă de expresii⟩ ⟨separator⟩ ⟨expresie⟩
⟨separator⟩ : : = , | ; ⟨terminator⟩ : : = , | ;
10 X$="DUMITRAŞCU"
20 V%=40
30 **LPRINT** X$, "LIVIU", V%

Enunţ. Valorile expresiilor din ⟨lista de expresii⟩ sînt tipărite la imprimantă potrivit tipului expresiei. Separatorul ' ; ' provoacă tipărirea membrilor listei unul după altul (întregii precedaţi de semn dacă sînt negativi; realii simplă precizie cu factor de scară dacă depăşesc 7 cifre; realii dublă precizie cu factor de scară dacă depăşesc 16 cifre). În cazul în care separatorul este ' ; ' tipărirea se face aliniat la subzone de 14 caractere. Dacă terminatorul e omis se forţează un ⟨CR⟩, în caz contrar efectul său va fi pentru enunţul **LPRINT** următor.

Cuvinte cheie asociate. **LPRINT USING, PRINT, PRINT USING, PRINT#, PRINT# USING**

LPRINT USING[(S)]

LPRINT USING ⟨format⟩; ⟨listă de expresii⟩ [⟨terminator⟩]
cu:
⟨listă de expresii⟩ : : = ⟨expresie⟩ | ⟨listă de expresii⟩ ⟨separator⟩ ⟨expresie⟩
⟨separator⟩ : : = , | ;
⟨terminator⟩ : : = , | ;
10 X$="ADELAIDA"
20 **LPRINT USING** "! " ; X$ 'SE VA TIPĂRI A
30 **LPRINT USING** "\   \ " ; X$ 'SE VA TIPĂRI ADELAI
40 Y=–67.366
50 **LPRINT USING** "–##·##"; Y 'SE VA TIPĂRI –67.37
60 Z%=–123582

70 **LPRINT USING** "– #### ^^^^" ; Z% 'SE VA TIPĂRI – 123 6 E+06

Enunț. Valorile expresiilor din ⟨lista de expresii⟩ sînt tipărite la imprimantă potrivit specificației de format ⟨format⟩. Aceasta este o expresie tip șir compusă din caractere de formatare.

a) pentru variabile tip șir:

! – se tipărește primul caracter

\ \ – se tipăresc cu 2 caractere mai mult ca numărul de spații între delimitatorii ' \ ' ; alinierea e la stînga

& – se tipărește întregul șir.

b) pentru variabile numerice:

: se tipărește cîte o cifră a numărului în poziția respectivă (cu aliniere dreapta).

· : indică poziția punctului zecimal.

+ : semnul numărului se plasează înainte sau după număr, funcție de poziția caracterului ' + ' față de specificatorul de format (la început sau la sfîrșit).

– : plasat după specificatorul de format, forțează înscrierea caracterului ' – ' în continuarea imaginii tipărite a numerelor negative.

** : plasate la începutul specificatorului de format, asigură umplerea spațiilor inițiale (rezultate din aliniere) cu caracterul ' * '; ele indică și poziția primelor două cifre.

$$: plasate la începutul specificatorului de format, asigură prezența caracterului ' $ ' la începutul numărului editat și în același timp poziția primei sale cifre.

**$: plasate la începutul specificatorului de format indică înlocuirea spațiilor inițiale (rezultate din aliniere) cu caracterul ' * ' și plasarea caracterului ' $ ' la începutul numărului, în același timp indicînd și poziția primelor 3 cifre.

, : plasată imediat înainte de punctul zecimal, provoacă separarea în forma editată, a triadelor prin caracterul ' , ' ; plasată la sfîrșitul specificatorului de format, provoacă forțarea în această poziție a caracterului ' , '.

^^^^ : plasată după șirul de caractere ' # ' ce indică șablonul de editare indică editarea sub formă exponențială.

– : caracterul imediat următor, considerat ca literal este transcris în șirul editat.

% : dacă numărul este mai mare decît șablonul ' # ' el se editează corect, dar precedat de caracterul ' % '.

Existența unui ⟨terminator⟩ provoacă tipărirea pe aceeași linie a următorului enunț tip **LPRINT** ori **LPRINT USING**.

Cuvinte cheie asociate. **LPRINT, PRINT, PRINT#, PRINT USING, PRINT# USING**

LSET(S)

LSET ⟨var șir⟩=⟨expresie șir⟩

100 **FIELD** #1, 2 **AS** NORD$, 33 **AS** NUME$

. . .

145 N%=**CVI** (NORD$)
150 **LSET** X$=**MKI$** (N%)

Enunț. Valoarea ⟨expresie șir⟩ este atribuită, aliniată la stînga, unei variabile șir ⟨var șir⟩ de sine stătătoare, sau care a fost definită de enunțul **FIELD** într-o zonă tampon pentru acces aleator.

Cuvinte cheie asociate. **FIELD MKD$, MKI$, MKS$, RSET**

MERGE(C)

MERGE ⟨specificator fișier⟩
MERGE "A : CAT 1 \ CAT 11 \ RUTIN"

Comandă. Programul GW–BASIC (în format ASCII) care este memorat pe disc prin ⟨specificator fișier⟩ este adus în memorie, liniile sale înlocuind liniile cu același număr de ordine ale programului deja existent, iar cele noi adăugîndu-se în secvență, la cele existente.

MID$(F)

MID$ (⟨șir⟩, ⟨start⟩ [,⟨lungime⟩])
10 X$="GW–BASIC"
20 **PRINT MID$** (X$,4), **MID$** (X$, 4, 3)
RUN
BASIC BAS

Funcție. Determină un subșir al expresiei șir ⟨șir⟩ care începe din poziția ⟨start⟩ (expresie numerică pozitivă cu valori ⩾1) și avînd lungimea maximă ⟨lungime⟩ (expresie aritmetică cu valori 0–256).

Dacă valoarea ⟨lungime⟩ depășește numărul de caractere rămase în șir din poziția ⟨start⟩ lungimea reală a subșirului va fi acest rezid.

Dacă ⟨lungime⟩ ia valoarea 0, se returnează șirul **NULL**.

Cuvinte cheie asociate. **LEFT$, LEN, RIGHT$**

MID$(S)

MID$ (⟨șir⟩, ⟨start⟩ [,⟨lungime⟩])=⟨subșir⟩
10 X$="ARTICOLE MOBILE"
20 **MID$** (X$, 6, 4)="ATII"
30 **PRINT** X$
RUN
ARTICULAȚII MOBILE

Enunț. Subșirul avînd lungimea ⟨lungime⟩ din șirul definit de expresia șir ⟨șir⟩ și începînd cu poziția ⟨lungime⟩ (expresie întreagă pozitivă cu valori ⩾1, implicit **LEN** (⟨subșir⟩) este înlocuit cu expresia șir ⟨subșir⟩ de aceeași lungime. Lungimea reală este de fapt min (⟨lungime⟩, rest șir începînd cu poziția ⟨poziție⟩).

Cuvinte cheie asociate. **LEFT$, LEN, RIGHT$**

MKDIR(C)

MKDIR ⟨traseu⟩
MKDIR "CATAL 1 \ CATAL 11"

Comandă. Crearea unui nou catalog la capătul căii ⟨traseu⟩ (expresie tip șir).

Cuvinte cheie asociate. **RMDIR**

MKD$(F)

MKD$ (⟨expresie reală dublă precizie⟩)
10 E#=7.5832
20 **FIELD** #1, 8 **AS** D$
30 **LSET** D$=**MKD$** (E#)

Funcţie. Converteşte ⟨expresie reală dublă precizie⟩ într-un şir de 8 octeţi.

Cuvinte cheie asociate. **CLD, CVI, CVS, MKI$, MK$**

MKI$(F)

MKI$ (⟨expresie întreagă⟩)
10 I%=17
20 **FIELD** #1, 2 **AS** A$
30 **LSET** A$=**MKI$** (I%)

Funcţie. Converteşte ⟨expresie întreagă⟩ într-un şir de 2 octeţi.

Cuvinte cheie asociate: **CVD, CVI, CVS, MKD$, MKS$**

MKS$(F)

MKS$ (⟨expresie reală simplă precizie⟩)
10 S=7.83
20 **FIELD** #1, 4 **AS** G$
30 **LSET** G$=**MKS$** (S)

Funcţie. Converteşte ⟨expresie reală simplă precizie⟩ într-un şir de 4 octeţi.

Cuvinte cheie asociate. **CVD, CVI, CVS, MKD$, MKI$**

NAME(C)

NAME ⟨specificator fişier 1⟩ **AS** ⟨specificator fişier 2⟩
NAME "\ CAT 1 \ CLIENŢI" **AS** "CATALOG\PRODUS\CLIENŢI"

Comandă. Schimbă ⟨specificator fişier 1⟩ al unui fişier închis cu ⟨specificator fişier 2⟩, fără a modifica însă şi specificarea dispozitivului.

NEW(C)

NEW

Comandă. Şterge programul curent, anulînd toate variabilele sale, închizînd în prealabil fişierele şi anulînd eventualele trasaje ale execuţiei.

Cuvinte cheie asociate. **TROFF**

OCT$(F)

OCT$ (⟨expresie numerică⟩)
PRINT OCT$ (32)
40

Funcţie. Calculează valoarea octală a argumentului ⟨expresie numerică⟩ (cu valori −32 768 ÷ 65 535).

Cuvinte cheie asociate. **HEX$**

ON COMn GOSUB(S)

OM COM (⟨n⟩) **GOSUB** ⟨număr linie⟩
225 **ON COM** (1) **GOSUB** 530

250 **COM** (1) **ON**
530 **REM** RUTINA TRATARE SOSIRE CARACTER ÎN TAMPON COM 1
— — — — — — — — — —
850 **RETURN** 310

Enunţ. Indică primul ⟨număr linie⟩ (expresie numerică) al unei subrutine de tratare a evenimentului provocat de sosirea caracterelor în zona tampon de comunicaţii a canalului ⟨n⟩ (expresie întreagă cu valori 1–4), dacă această opţiune e activată printr-un enunţ **COM** (⟨n⟩) **ON** anterior. De notat că întreruperea e dezactivată dacă expresia ⟨număr linie⟩ are valoarea 0. La activarea rutinei se execută un **COM** (⟨n⟩) **STOP** implicit, iar la ieşirea din ea (cu **RETURN**) se execută implicit un enunţ **COM** (⟨n⟩) **ON** (exceptînd execuţia explicită în rutină a unui enunţ **COM** (⟨n⟩) **OFF.**

Cuvinte *cheie asociate*. **COMn OFF, COMn ON, COMn STOP**

ON ERROR GO TO(S)

ON ERROR GO TO ⟨număr linie⟩
10 **ON ERROR GOTO** 100
20 **INPUT** N
30 M=**LOG**(N) 'DACĂ ARGUMENTUL E NEGATIV APARE EROARE 5
.
.
.
100 **REM** 'RUTINA TRATARE EROARE
110 **IF ERR**=5 **THEN PRINT** 'NUMĂR NEGATIV – REPETAŢI" :
RESUME 20

Enunţ. Dacă expresia numerică întreagă ⟨număr linie⟩ este nenulă se activează controlul de către utilizator a unei erori GW–BASIC printr-o rutină de întrerupere care începe cu linia de rang ⟨număr linie⟩. Dacă expresia ⟨număr linie⟩ are valoarea 0, tratarea întreruperii programului prin rutină utilizator este inhibată, GW–BASIC afişînd un mesaj de eroare şi oprind execuţia.

Într-o rutină utilizator de această speţă nu sînt permise alte întreruperi de tip eroare GW–BASIC.

Observaţie. Orice rutină de tratare permite revenirea în programul GW–BASIC cu enunţul **RESUME**.

Cuvinte cheie asociate. **ERL, ERR, ERROR, ON COMn, GOSUB, RESUME**

ON . . . GOSUB(S)

ON ⟨expresie numerică⟩ **GOSUB** ⟨listă numere linii⟩
cu
⟨listă numere linii⟩ : : = ⟨număr linie⟩ | ⟨liste numere linie⟩,
⟨număr linie⟩
65 **REM** EXECUTAREA UNEIA DIN SUBRUTINELE 1001,
1002, 1003 FUNCŢIE DE VALOAREA LUI I%
70 **ON** I% **GOSUB** 1001, 1002, 1003

Enunţ. Permite executarea subrutinei GW–BASIC de rang ⟨expresie numerică⟩ care începe cu linia ⟨număr linie⟩ din ⟨lista numere linii⟩.

Dacă valoarea ⟨expresie numerică⟩ este 0 sau depăşeşte cardinalul ce corespunde la ⟨lista numere linii⟩ execuţia continuă în secvenţă.

Cuvinte cheie asociate. **ON . . . GOTO**

ON . . . GOTO(S)

ON ⟨expresie numerică⟩ **GOTO** ⟨lista numere linii⟩
cu
⟨lista numere linii⟩ : : = ⟨număr linie⟩ | ⟨lista numere linii⟩, ⟨număr linie⟩

```
        . . .
65 REM SE EXECUTĂ SALTUL LA LINIILE 1001, 1002, 1003
66 REM FUNCŢIE DE VALORILE LUI I%
70 ON I% GOTO 1001, 1002, 1003
        . . .
```

Enunţ. Se transferă controlul la linia GW–BASIC de rang ⟨expresie numerică⟩ care începe cu linia ⟨număr linie⟩ din ⟨lista numere linii⟩.

Dacă valoarea ⟨expresie numerică⟩ este 0 sau depăşeşte cardinalul ce corespunde la ⟨lista numere linii⟩, execuţia continuă în secvenţă.

Cuvinte cheie asociate. **ON . . . GOSUB**

ON KEYn GOSUB(S)

ON KEY (⟨n⟩) **GOSUB** ⟨număr linie⟩

```
10 REM CHEIA UTILIZATOR 15 E PROGRAMATĂ PT. TASTARE CTRL/A
15 REM CHEIA UTILIZATOR 16 E PROGRAMATĂ PT. TASTARE CTRL/B
20 KEY 15 CHR$ (&H04)+CHR$ (30)
25 KEY 16, CHR$ (&H04)+CHR$ (48)
30 KEY (15) ON'ACTIVITATE ÎNTRERUPERE PENTRU CTRL/A (KEY 15)
35 KEY (16) ON'ACTIVARE ÎNTRERUPERE PT. CTRL/B (KEY 16)
40 PRINT "PROGRAMUL CICLEAZĂ; CU CTRL/A AFIŞAŢI ORA"
45 PRINT "CU CTRL/B ÎNCHEIAŢI EXECUŢIA"
50 ON KEY (15) GOSUB 100
55 ON KEY (16) GOSUB 200
60 I%=1
65 J%=17
70 ON I% GOTO 65, 75
75 END
100 PRINT DATE$
110 RETURN
200 I%=2 'MODIFICĂ VARIABILA CONTROL CICLU
210 RETURN
```

Enunţ. Indică ⟨număr linie⟩ (expresie numerică) de debut a unei subrutine de tratare a evenimentului reprezentat de acţionare a cheii avînd valoarea expresiei numerice ⟨n⟩

– chei funcţionale F1–F10 (⟨n⟩=1–10)
– cursor în sus (⟨n⟩=11)
– cursor la stînga (⟨n⟩=12)
– cursor la dreapta (⟨n⟩=13)
– cursor în jos (⟨n⟩=14)
– chei definite de utilizator (⟨n⟩=15)

în condițiile în care întreruperea este activată prin enunțul **KEY** (⟨n⟩) **ON** correspondent.

Dacă ⟨număr linie⟩ ia valoarea 0, întreruperea e inhibată.

În timpul activării rutinei de întrerupere se execută un enunț **KEY** (⟨n⟩) **STOP** implicit, în vreme ce la ieșirea din rutină, se execută un enunț **KEY** (⟨n⟩) **ON** implicit (exceptînd specificarea expresă în corpul rutinei a unui enunț **KEY** (⟨n⟩) **OFF**).

Cuvinte cheie asociate. **KEYn OFF, KEYn ON, KEYn STOP**

ON PLAYn GOSUB(S)

ON PLAY (⟨n⟩) **GOSUB** ⟨număr linie⟩

```
10 PLAY ON
20 ON PLAY (6) GOSUB 150
.
.
.
150 REM ÎNTRERUPERE DIMINUARE TAMPON MELODII
250 RETURN
```

Enunț. Indică ⟨număr linie⟩ (expresie numerică întreagă) de debut al unei subrutine de tratare a evenimentului provocat de diminuarea numărului de note din tamponul de melodii sub ⟨n⟩ (expresie întreagă cu valori 1–32) note, în condițiile în care întreruperea a fost activată printr-un enunț **PLAY ON** și dacă se execută o melodie de fond.

Dacă ⟨număr linie⟩ ia valoarea 0, întreruperea este inhibată.

În timpul activării rutinei de întrerupere se execută un enunț **PLAY STOP** implicit, în vreme ce la ieșirea din rutină se execută un enunț **PLAY ON** implicit (exceptînd specificarea expresă în cadrul rutinei a unui enunț **PLAY OFF**).

Cuvinte cheie asociate. **PLAY, PLAY OFF, PLAY ON, PLAY STOP**

ON TIMERn GOSUB(S)

ON TIMER (⟨n⟩) **GOSUB** ⟨număr linie⟩

```
10 REM SE MARCHEAZĂ OROLOGIUL CURENT DIN 3 ÎN 3 MINUTE
20 ON TIMER (180) GOSUB 500
30 TIMER ON
.
.
.
500 SAVROW=CSRLIN 'MEMORARE RÎND CURENT
510 SAVCOL=POS (0) 'MEMORARE COLOANĂ CURENTĂ
520 LOCATE 23, 20 'POZIȚIONARE PE LINIA 23 COLOANA 20
530 PRINT TIME$ 'IMPRIMARE OROLOGIU
540 LOCATE SAVROW, SAVCOL 'REPOZIȚIONARE CURSOR
550 RETURN
```

Enunț. Indică ⟨număr linie⟩ (expresie întreagă) de debut al unei subrutine de tratare a evenimentului provocat la parcurgerea unui interval de ⟨n⟩ (expresie numerică cu valori 1–86400) secunde, dacă întreruperea a fost activată printr-un enunț **TIMER ON** anterior.

În timpul activării rutinei de întrerupere se execută un enunț **TIMER STOP** implicit, în vreme ce la ieșirea din rutină se execută un enunț **TIMER**

ON implicit (exceptînd specificarea expresă în cadrul rutinei a unui enunţ **TIMER OFF**).

Cuvinte cheie asociate. **TIMER OFF, TIMER ON, TIMER STOP**

OPEN(S)

OPEN ⟨specificator fişier⟩ [**FOR** ⟨mod 1⟩] **AS** [#] ⟨număr canal⟩ [**LEN**=⟨lungime înregistrare⟩]

sau

OPEN ⟨mod 2⟩, [#] ⟨număr canal⟩, ⟨specificator fişier⟩ [,⟨lungime înregistrare⟩]

a)
```
10 REM DESCHIDE FIŞIER PENTRU INTRARE FORMAT 1
20 OPEN "\CATAL 1\CATAL 11\STOC. DAT" FOR INPUT AS #3
```
cu varianta:

b)
```
10 REM DESCHIDE FIŞIER PENTRU INTRARE – FORMAT 2
20 OPEN "I", #3, "\CATAL 1\CATAL 11\STOC. DAT"
```
c)
```
10 REM DESCHIDERE IMPRIMANTĂ CU DRIVER GW–BASIC
20 OPEN "LPT:" FOR OUTPUT AS 2
```
cu varianta

d)
```
10 REM DESCHIDERE IMPRIMANTĂ CU DRIVER MS–DOS
20 OPEN "\DEV\LPT1" FOR OUTPUT AS 2
```

Enunţ. Permite deschiderea (în sensul sistemului de operare) a fişierului desemnat extern prin ⟨specificator fişier⟩ şi intern prin ⟨număr canal⟩ (expresie întreagă cu valori 1–255), număr asociat de fapt zonei tampon de intrare/ieşire aparţinînd acestui fişier. Dacă fişierul este exploatat aleator se poate indica şi cîte o ⟨lungime înregistrare⟩ (expresie întreagă cu valori 1–32767, dar nu mai mare ca valoarea comutatorului /S: ⟨recsiz⟩ de la lansarea sub MS–DOS a interpretorului GW–BASIC).

Fişierul este deschis potrivit ⟨mod1⟩ (cuvînt cheie) sau ⟨mod2⟩ (expresie şir din a cărei valoare doar primul caracter e relevant):

⟨mod 1⟩	*⟨mod 2⟩*	Semnificaţie
INPUT	"I"	acces secvenţial la intrare, cu poziţionare pe începutul fişierului existent
OUTPUT	"O"	acces secvenţial la ieşire, cu poziţionare pe începutul fişierului existent, eventual creat cu această ocazie
APPEND	"A"	acces secvenţial la ieşire cu poziţionare pe sfîrşitul fişierului existent, eventual creat cu această ocazie
–	"R"	acces aleator

Observaţii

a) Numărul maxim de fişiere deschise simultan într-un program GW–BASIC nu poate depăşi valoarea comutatorului /F: ⟨număr fişiere⟩ de la lansarea sub MS–DOS a interpretorului GW–BASIC (implicit 3).

b) Semnificaţia accesului aleator pentru dispozitivele de ieşire LPT1 :, LPT2 :, LPT3 : este aceea că fiecare ⟨CR⟩ nu va fi însoţit automat şi de un ⟨LF⟩. Lungimea înregistrării dispozitivelor orientate spre caractere este 1.

c) Informaţiile de control pentru dispozitive/fişiere sînt scrise de enunţul GW–BASIC IOCTL şi citite cu funcţia GW–BASIC IOCTL$.

d) Deşi unul şi acelaşi fişier fizic poate fi deschis ca fişier secvenţial de intrare sau cu acces aleator utilizînd simultan mai multe numere de canal, acest lucru este interzis pentru fişiere secvenţiale de ieşire sau în extensie.

Cuvinte cheie asociate. **IOCTL, IOCTL$, OPEN COM**

OPEN COM(S)

OPEN "COM ⟨n⟩ : [⟨viteza⟩] [,⟨paritate⟩][,[⟨bit dat⟩] [,[⟨stop⟩] [**,RS**] [**,CS**[⟨t⟩]] [**,DS**[⟨t⟩]] [**,BIN**] [**,ASC**] [**,LF**]]]] "[**FOR**⟨mod⟩ **AS #** ⟨număr canal⟩ [LEN=⟨lungime înregistrare⟩]

10 **REM** VITEZA 1200 BAUDS, FĂRĂ PARITATE, 7 BITS DATE, 1 BIT STOP MOD BINAR
20 **OPEN** "COM 1 : 1200, N, 7, 1, BIN" **AS** #1

Enunţ. Se deschide şi iniţializează liniile de comunicaţie pentru operaţiile de intrare/ieşire pentru dispozitivul de comunicaţie ⟨n⟩ (expresie întreagă cu valori 1–4), eventual stabilindu-se o serie de parametri, cum ar fi ⟨viteza⟩ (constantă întreagă cu valori 75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, implicit 300) avînd o ⟨paritate⟩ (literal cu valori: N – fără paritate, E – paritate pară, M – marcă, O – paritate impară, S – spaţiu, implicit 6) utilizînd un protocol cu un număr de biţi de date ⟨bit dat⟩ (expresie întreagă cu valori 5–8, implicit 7) şi ⟨stop⟩ (expresie numerică cu valori 1, 1. 5, 2, implicit 1 pentru viteze de la 150 bauds în sus). În plus protocolul poate prevedea o serie de opţiuni.

RS – suprimare RTS
CS [⟨t⟩] – controlare CTS
cu
⟨t⟩ – milisecunde 'time-out' (implicit 13)
DS [⟨t⟩] – controlare DSR (Data send Ready)
cu
⟨t⟩ – milisecunde 'time-out' (implicit 1000)
CD [⟨t⟩] – controlează CD (Curier Detect)
cu
⟨t⟩ – milisecunde 'time-out' (implicit 1000)
BIN – deschidere în mod binar (implicit)
ASC – deschidere în mod caracter (ASCII)
LF – indică trimiterea unui ⟨LF⟩ după un ⟨CR⟩ (opţiune anulată de opţiunea BIN)

Din punct de vedere GW–BASIC dispozitivului de comunicaţii i se asociază un ⟨număr canal⟩, iar deschiderea se face potrivit unui ⟨mod⟩, reprezentat de cuvinte cheie.

INPUT – fişiere secvenţiale de intrare
OUTPUT – fişiere secvenţiale de ieşire
(fără indicare ⟨mod⟩) – fişiere accesate aleator

Operaţiile de intrare/ieşire se desfăşoară pe o ⟨lungime înregistrare⟩ (expresie întreagă cu valori implicite de 2 pentru zone tampon de emisie şi 128 pentru zone tampon recepţie) ce nu pot depăşi valoarea Comutatorului /C: ⟨buf siz⟩ din comanda MS–DOS de activare a interpretorului GW–BASIC.

Cuvinte cheie asociate. **OPEN**

OPTION BASE(S)

OPTION BASE ⟨n⟩
20 **DIM** A%(10)

```
30 OPTION BASE 1 'ELEMENTELE LUI A% VOR FI A% (1) . . A% (10)
40 FOR I%=1 TO 10 'SE INIŢIALIZEAZĂ ACESTE ELEMENTE
50 A% (I%)=I%
60 NEXT I%
```

Enunţ: Defineşte prin valoarea lui ⟨n⟩ (expresie întreagă valori 0–1) limită inferioară pentru indicii de tablouri (implicit 0).

Cuvinte cheie asociate. **DIM**

OUT(S)

```
OUT ⟨port⟩, ⟨octet⟩
10 OUT 135, 254
```

Enunţ. Se transmite la o linie de comunicaţii avînd numărul ⟨port⟩ (expresie întreagă pozitivă cu valori 0–65535) un caracter de cod ASCII reprezentat de numărul ⟨octet⟩ (operaţie întreagă pozitivă cu valori 0–255).

PAINT(SGR)

```
PAINT [STEP] (⟨x⟩, ⟨y⟩) [;⟨tapet⟩] [,⟨contur⟩] ⟨fond⟩]]
10 SCREEN 1 'ALEGERE MOD REZOLUŢIE MEDIE
20 COLOR 0, 0, 1, 0 'FOND TEXT NEGRU,
   PALETĂ VERDE–ROŞU–GALBEN
25 REM DESENARE CU VERDE PE FOND GRAFIC NEGRU
30 CLS 'ŞTERGERE ECRAN PENTRU PREGĂTIRE FIGURĂ
40 CIRCLE (256, 128) 130, 2 'CERC CU ROŞU (CULOARE 2)
50 PAINT (256, 128) 1, 2 'UMPLUT CU VERDE
60 LINE (251, 123) – STEP (10, 10), O, BF 'DREPTUNGHI NEGRU
   ÎN CENTRU CERC
```

Enunţ: Asigură colorarea sau mozaicarea unei arii grafice cu o culoare definită de ⟨tapet⟩ (expresie numerică cu valori 0–3) sau cu un model definit tot de ⟨tapet⟩, care de această dată este reprezentat de o expresie de tip şir care descrie acest şablon. Şablonul este reprezentat din 1–64 octeţi care reprezintă măştile a 8 bits. Automat are loc rotarea bits mască pentru a asigura alinierea pe axa y după formula:

bit mască – şablon=y **MOD** lungime şablon

Aşa-zisul model al fondului de mozaic necesar a fi sărit cînd se testează terminarea conturului (frontierei) este dat de ⟨fond⟩ (expresie şir, cu valoare implicită **CHR$** (0). Această opţiune este necesară dacă zona are aceeaşi culoare ca şi două linii consecutive din şablon.

Figura de umplut are o frontieră a cărei culoare se trasează cu ⟨contur⟩ (expresie numerică întreagă cu valori 0–3, implicit valoarea ⟨tapet⟩). Propriu-zis vopsirea/mozaicarea încep din poziţiile de abscisă ⟨x⟩ şi ordonată ⟨y⟩ (expresii numerice) considerate în coordonate absolute sau relative (opţiune **STEP** prezentă).

Cuvinte cheie asociate. **COLOR**.

PEEK(F)

```
PEEK (⟨ecart⟩)
A=PEEK (&H5A00)
```

Funcţie. La o deplasare ⟨ecart⟩ (expresie numerică întreagă cu valori –32768–65535) faţă de segmentul curent definit de enunţul **DEF SEG** se găseşte adresa unui octet din memorie, al cărui conţinut este furnizat utilizatorului.

Cuvinte cheie asociate. **DEF SEG, POKE, VARPTR.**

PLAY[(S)]

PLAY ⟨expresie şir⟩
10 **LET** X$="T 180 02 P2 P8 L8 GGG L2 E–"
20 **LET** Y$="P24 P8 L8 FFF L2 D"
30 **LET PLAY** X$+Y$

Enunţ. Se execută melodia reprezentată de caracterele din ⟨expresie şir⟩. Şirul conţine şi monoinstrucţiuni muzicale, alcătuite din comenzi specifice.

A–G[#|+|–] –	Se cîntă notele din gama A–G (Do–Si) cu note ascuţite (sufixe "#" ori 'T') sau plate (sufix '–').
O⟨n⟩–	Stabilirea octavei curente la ⟨n⟩ (expresie întreagă cu valori 0–6).
>⟨n⟩ –	Trecere în octavă superioară la fiecare execuţie a notei ⟨n⟩, pînă ce s-a ajuns în octava 6 care rămîne octava notei.
<⟨n⟩ –	Trecerea în octava inferioară la fiecare execuţie a notei ⟨n⟩ pînă ce s-a ajuns în octava 0 care rămîne octava notei.
N⟨n⟩ –	În cele 7 octave posibile se execută una din 84 note nominalizate cu ⟨n⟩ (expresie întreagă; 0 înseamnă pauză).
P⟨n⟩ –	Specifică o pauză pentru nota ⟨n⟩ (expresie întreagă cu valori 1–84) de durata dată de comanda L.
L⟨n⟩ –	Lungimea fiecărei note ce urmează este ⟨n⟩ (expresie întreagă cu valori 1–64, semnificînd inversul valorii notei).
–	Scalarea cu 3/2 a duratei notei ce urmează (faţă de cea din comanda L).
T⟨n⟩ –	Tempoul (numărul de 4 note pe minut) este stabilit la ⟨n⟩ (expresie întreagă cu valori 32–255, implicit 120).
MB –	Stabilirea calificativului de muzică de fond, atît pentru enunţul **PLAY**, cît şi pentru enunţul **SOUND**.
MF –	Stabilirea calificativului de muzică efectivă (executată notă cu notă).
MN –	Stabilirea caracterului de normalitate pentru muzică, deci fiecare notă executată într-un timp de 7/8 din cel specificat de comanda L.
ML –	Muzică 'legato', deci toată durata dată de comanda L.
MS –	Muzică 'staccato', fiecare notă se execută la 3/4 din durata indicată de comanda L.
X⟨şir⟩ –	Se execută un ⟨şir⟩.

Cuvinte cheie asociate. **PLAYn, PLAY OFF, PLAY ON, PLAY STOP**

PLAYn[(F)]

PLAY (⟨argument fictiv⟩)
IF PLAY (0)=5 **GOTO** 310

Funcție. Se furnizează numărul de note din tamponul melodiei de fond. Argumentul e fictiv și precizat doar pentru păstrarea compatibilității cu caracterul de funcție. Valoarea e 0 dacă muzica executată este efectivă.

Cuvinte cheie asociate. **PLAY, PLAY OFF, PLAY ON, PLAY STOP.**

PLAY OFF (S)

PLAY OFF

Enunț. Inhibă întreruperea provocată de atingerea unei limite în conținutul zonei tampon de melodii de fond.

Cuvinte cheie asociate. **ON PLAYn GOSUB, PLAYn, PLAY ON, PLAY STOP**

PLAY ON (S)

PLAY ON

Enunț. Activează întreruperea provocată de atingerea unei limite în conținutul zonei tampon de melodii de fond.

Cuvinte cheie asociate. **ON PLAYn GOSUB, PLAYn, PLAY OFF, PLAY STOP**

PLAY STOP (S)

PLAY STOP

Enunț. Suspendă întreruperea provocată de atingerea unei limite în conținutul zonei tampon de melodii de fond.

Cuvinte cheie asociate. **ON PLAYn GOSUB, PLAYn, PLAY OFF**

PMAP (FGR)

PMAP (⟨coordonată⟩, ⟨n⟩)

```
    SCREEN 2
    WINDOW SCREEN (80, 100)-(200, 200)
    X1=PMAP (80, 0)
    0
OK
    Y1=PMAP (200, 1)
    199
OK
    X2=PMAP (619, 2)
    199
OK
    Y2=PMAP (100, 3)
OK
    140
```

Funcție. Convertește valoarea ⟨coordonată⟩ (expresie numerică) desemnînd abscisa sau ordonata unui punct în conformitate cu opțiunea ⟨n⟩ care e o expresie numerică întreagă cu valorile:

0 – ⟨coordonata⟩ este o abscisă în coordonate univers, pentru care se determină echivalentul fizic.

1 – ⟨coordonata⟩ este o ordonată în coordonate univers, pentru care se determină echivalentul fizic.

2 – ⟨coordonata⟩ este o abscisă în coordonate fizice pentru care se determină echivalentul univers.

3 – ⟨coordonata⟩ este o ordonată în coordonate fizice pentru care se determină echivalentul univers.

Cuvinte cheie asociate. **SCREEN, VIEWPORT SCREEN, WINDOW SCREEN**

POINT (F)

```
POINT (⟨arg⟩)
cu ⟨arg⟩ : : =⟨x⟩, ⟨y⟩ | ⟨n⟩
200 SCREEN 1 'Rezoluţie medie
205 K%=0
210 FOR I=1 TO 320
220   FOR J=1 TO 200
230     IF POINT (I, J)=3 THEN K%=K%+1
240 NEXT J, I
250 PRINT "CULOAREA 3 ESTE POSEDATĂ DE"; K%; "PUNCTE"
```

Funcţie. Dacă funcţia are două argumente, se determină culoarea elementului de imagine avînd coordonatele absolute ⟨x⟩ şi ⟨y⟩ (expresii întregi). Valoarea funcţiei este –1 dacă cel puţin una din coordonate nu e în domeniul ecranului.

Dacă funcţia are un singur argument, se determină punctul grafic curent, funcţie de valoarea expresiei întregi ⟨n⟩:

0 – abscisa fizică curentă
1 – ordonata fizică curentă
2 – abscisa curentă în coordonate univers, dacă există un enunţ **WINDOW** anterior (altminteri acelaşi efect ca **POINT** (0))
3 – ordonata curentă în coordonate univers, dacă există un enunţ **WINDOW** anterior (altminteri acelaşi efect ca **POINT** (1))

POKE (S)

```
POKE ⟨ecart⟩, ⟨octet⟩
15 POKE &H5A00, &HFA 'PLASARE valoare 250 la adresa segment
                      curent+5A00H
```

Enunţ. Caracterul avînd codul ⟨octet⟩ (expresie numerică pozitivă cu valori 0–255) este înscris la adresa dată de o deplasare ⟨ecart⟩ (expresie numerică pozitivă cu valori 0–65535) faţă de segmentul curent definit de un enunţ **DEF SEG** anterior.

Cuvinte cheie asociate. **DEF SEG, PEEK.**

POS (F)

```
POS (⟨argument fictiv⟩)
IF POS (0)>72 THEN BEEP 'SONERIE DACĂ CURSORUL A DEPĂŞIT
                        COLOANA 72
```

Funcţie. Determină abscisa curentă a cursorului (valori în intervalul 1–40, sau 1–80). Argumentul e fictiv şi prevăzut pentru compatibilitate cu caracterul de funcţie.

Cuvinte cheie asociate. **CSRLIN.**

PRESET(SGR)

PRESET [**STEP**](⟨x⟩, ⟨y⟩)[,⟨culoare⟩]
PRESET (15, 60), 3

Enunț. Se desenează pe ecran cu un număr ⟨culoare⟩ (expresie întreagă cu valori 0–3, implicit cea de fond) un element imagine corespunzînd 'punctului' de abscisă ⟨x⟩ și ordonată ⟨y⟩ (expresii numerice întregi) în coordonate absolute (sau relative dacă opțiunea **STEP** e prezentă).

Cuvinte cheie asociate. **COLOR, PSET.**

PRINT (S)

PRINT [⟨listă de expresii⟩][⟨terminator⟩]
cu:
⟨lista de expresii⟩ : : = ⟨expresie⟩ | ⟨listă de expresii⟩ ⟨separator⟩ ⟨expresie⟩
⟨separator⟩ : : = , | ;
⟨terminator⟩ : = , | ;

```
10 X$="DUMITRAŞCU"
20 V%=40
30 PRINT X$, "LIVIU", V%
RUN
DUMITRAŞCU LIVIU 40
```

Enunț. Valorile expresiilor din ⟨lista de expresii⟩ sînt afișate pe ecran. Separatorul ' ; ' provoacă tipărirea membrilor listei unul după altul (întregii precedați de semn dacă sînt negativi; realii simpli precizie cu factor de scară dacă depășesc 7 cifre: realii dublă precizie cu factor de scară dacă depășesc 16 cifre). În cazul în care separatorul este ' , ' tipărirea se face aliniat la subzone de 14 caractere. Dacă terminatorul e omis se forțează un ⟨**CR**⟩, în caz contrar efectul său va fi pentru enunțul **PRINT** următor.

Cuvinte cheie asociate. **LPRINT, LPRINT USING, PRINT USING, PRINT #, PRINT # USING**

PRINT USING (S)

PRINT USING ⟨format⟩; ⟨listă de expresii⟩ [⟨terminator⟩]
cu ⟨listă de expresii⟩ : : = ⟨expresie⟩ | ⟨listă de expresii⟩ ⟨separator⟩⟨expresie⟩
⟨separator⟩ : : = , | ;
⟨terminator⟩ : : = , | ;

```
10 X$="ADELAIDA"
20 PRINT USING "!"; X$ 'SE VA TIPĂRI A
30 PRINT USING "\ \" X$ 'SE VA TIPĂRI ADELAI
40 Y=-67.366
50 PRINT USING "-##.##"; 'SE VA TIPĂRI -67.37
60 Z%=-123582
70 PRINT USING "-.####^^^^"; Z% 'SE VA
                                        TIPĂRI -.1236E+06
RUN
A
ADELAI
-67.37
-.1236E+06
```

Enunţ. Valorile expresiilor din ⟨lista de expresii⟩ sînt afişate pe ecran potrivit specificaţiei de format ⟨format⟩.

Aceasta este o expresie tip şir compusă din caractere de formatare.

Cuvinte cheie asociate. **LPRINT, LPRINT USING, PRINT, PRINT#, PRINT#USING**

PRINT# (S)

PRINT#⟨număr canal⟩, ⟨listă de expresii⟩

cu

⟨lista de expresii⟩ : : = ⟨expresie⟩ | ⟨listă de expresii⟩; ⟨expresie⟩

```
100 X$="POPESCU, LAURA"
110 Y$="20 ANI"
115 REM CODUL PENTRU "ESTE 34
120 PRINT#1, CHR$(34); X$; CHR$(34); CHR$(34); Y$; CHR$(34)
.
.
.
125 REM IN A$ ŞI B$ SE VOR CITI CORECT VECHILE X$ RESPECTIV
    Y$
130 INPUT#1,A$,B$
```

Enunţ. Valorile expresiilor din ⟨lista de expresii⟩ sînt scrise în fişierul de ieşire asociat prin enunţul **OPEN** corespunzător la numărul GW-BASIC ⟨număr canal⟩. Se recomandă utilizarea separatorului " ; " între expresii. Mai mult chiar, pentru a regăsi ulterior cu enunţuri **INPUT** aceste înregistrări se recomandă introducerea unor separatori expliciţi incluşi între perechile de ghilimele " ", sau chiar prin funcţia **CHR$**.

Cuvinte cheie asociate. **LPRINT, LPRINT USING, OPEN, PRINT, PRINT USING, PRINT#USING, WRITE#.**

PRINT# USING (S)

PRINT# ⟨număr canal⟩, **USING** ⟨format⟩; ⟨listă de expresii⟩

cu ⟨listă de expresii⟩ : : =⟨expresie⟩ | ⟨listă de expresii⟩; ⟨expresie⟩

```
100 I%=11 : K%=17 : L%=14
110 PRINT#1, USING "$$###.##."; I%; K%; L%.
```

Enunţ. Valorile expresiilor din ⟨lista de expresii⟩ sînt scrise în fişierul de ieşire asociat prin enunţul **OPEN** corespunzător la numărul GW-BASIC ⟨număr canal⟩, potrivit specificaţiei de format ⟨format⟩. Aceasta este o expresie tip şir compusă din caractere de formatare. Se recomandă utilizarea separatorului " ; " între expresii, precum şi utilizarea corectă a altor separatori pentru a permite regăsirea corectă a valorilor prin enunţuri ulterioare de tip **INPUT**.

Cuvinte cheie asociate. **LPRINT, LPRINT USING, OPEN, PRINT, PRINT USING, PRINT#**

PSET (SGR)

PSET [**STEP**] (⟨x⟩, ⟨y⟩) [,⟨culoare⟩]

```
10 REM SE SIMULEAZĂ EFECTUL LUI LINE, DAR FĂRĂ INTERPOLARE
```

```
20 FOR I=0 TO 100
30    PSET (I, I)
40 NEXT I
```

Enunț. Provoacă iluminarea cu un număr ⟨culoare⟩ (expresie întreagă cu valori 0–3, implicit 'cerneala' grafică) a elementului de imagine de abscisă ⟨x⟩ și ordonată ⟨y⟩ (expresii întregi), coordonatele fiind absolute (sau relative dacă se folosește opțiunea **STEP**).

Cuvinte cheie asociate. **COLOR, PRESET.**

PUT (SCOM)

```
PUT [#] ⟨număr canal⟩ [,⟨lungime⟩]
50 PUT #1, 80 'Scrie 80 octeți în tamponul de comunicații.
```

Enunț. Scrie în tamponul de comunicații asociat numărului GW-BASIC ⟨număr canal⟩ (prin enunțul **OPEN COM**) un număr de ⟨lungime⟩ octeți (expresie întreagă pozitivă cu valoare nedepășind valoarea precizată de parametrul **LEN** al comenzii **OPEN COM**).

Cuvinte cheie asociate. **OPEN COM.**

PUT (SFILE)

```
PUT [#]⟨număr canal⟩[,⟨număr înregistrare⟩]
10 OPEN "R", #3, "\CATAL 1\CATAL 11\STOC. DAT" 33
20 FIELD #3, 5 AS CODMAT$, 20 AS DENMAT$, 8 AS STOC$
30 FOR I=1 TO 10
40    INPUT "COD MATERIAL"; COD$
50    INPUT "DENUMIRE MATERIAL"; DEN$
60    INPUT "STOC XXXX.YYY"; ST
70    LSET CODMAT$=COD$
80    LSET DENAMT$=DEN$
90    LSET STOC$=MKS$(ST)
100   PUT #3, I
110 NEXT I
```

Enunț. Înregistrarea de rang ⟨număr înregistrare⟩ (expresie întreagă pozitivă cu valori 1–16777215, implicit cel curent în sens MS–DOS) prezentă într-o zonă tampon directă este înscrisă în fișierul de ieșire exploatat în mod aleator căruia i s-a asociat numărul GW-BASIC ⟨număr canal⟩ printr-un enunț **OPEN** corespunzător.

Cuvinte cheie asociate. **FIELD, LSET, OPEN, RSET.**

PUT (SGR)

```
PUT (⟨x⟩,⟨y⟩),⟨masiv⟩[,⟨verb⟩]
```

Enunț. Se transferă pe ecran într-un dreptunghi al cărui colț NV este dat de punctul de abscisă ⟨x⟩ și ordonată ⟨y⟩ (expresii numerice pozitive) imaginea stocată în tabloul de octeți ⟨masiv⟩ (tip numeric), interacționîndu-se cu imaginea deja prezentă pe ecran, potrivit modului stabilit de cuvîntul cheie ⟨verb⟩ ce are valorile:

PSET – transfer punct cu punct, cu respectarea culorii din momentul salvării imaginii (cu enunțul **GET**);

PRESET – se transferă punct cu punct reversul imaginii stocate în momentul salvării sale (cu enunțul **GET**);

AND – se efectuează operația logică și între elementul de imagine existent pe ecran și cel salvat în ⟨masiv⟩;

OR – se efectuează o supraimprimare a imaginii salvate pe cea existentă (operația logică SAU);

XOR – se efectuează o operație logică SAU exclusiv între elementul de imagine curent și cel salvat (permițînd eventual restaurarea imaginii anterioare și deci un gen de animație).

Cuvinte cheie asociate. **GET**

RANDOMIZE (S)

RANDOMIZE [⟨expresie numerică⟩]

```
10 RANDOMIZE
20 FOR I=1 TO 5
30   PRINT RND;
40 NEXT I
RUN
Random Number Seed (-32768 to 32767)?4
.628988 .765605 .5551561 .795727 .7834911.
```

Enunț. Se restabilește valoarea de pornire ('sămînța') generatorului de numere pseudoaleatoare la ⟨expresie numerică⟩. În lipsa argumentului, el va fi furnizat la momentul execuției pe baza mesajului GW-BASIC

Random Number Seed (−32768 to 32767)?

Valoarea furnizată static/dinamic va fi luată în considerare la invocarea enunțului **RND**.

Cuvinte cheie asociate. **RND.**

READ (S)

READ ⟨listă de variabile⟩

```
10 DIM M%(10)
20 OPTION BASE 1
30 REM CITIREA INIȚIALĂ A NUMERELOR DE SORTAT ÎN MASIV M%
40 FOR I=1 TO 10
50   READ M%(I)
60 NEXT I
70 DATA 4, 10, 2, 8, 3
80 DATA 7, 5, 1, 6, 9
   .
   .
   .
```

Enunț. Fiecărei variabile din ⟨lista de variabile⟩ i se atribuie o valoare corespunzătoare dintr-unul sau mai multe blocuri definite de enunțul **DATA**, începînd cu poziția curentă, cu respectarea strictă a concordanței de tip. Restabilirea acestei poziții pe începutul blocului se face cu un enunț **RESTORE**.

Cuvinte cheie asociate. **DATA, RESTORE**

REM (S)

REM ⟨comentarii⟩

Enunț. Nu este un enunț executabil, permite doar inserarea unor linii de ⟨comentarii⟩.

Observație. Orice enunț permite comentarea sa sub forma

[⟨număr linie⟩]⟨enunț de bază⟩'⟨comentarii⟩

RENUM(C)

RENUM [⟨număr linie nou⟩][,⟨număr linie vechi⟩][,⟨increment⟩]]
RENUM 'Renumerotarea tuturor liniilor din program în secvențe 10, 20, 30 . . .
RENUM 250, , 20 'Renumerotează liniile programului în secvențe 250, 270 . . .
RENUM 400, 350, 15 'Liniile de la 350 în sus sînt renumerotate cu 350, 365, 380. . .

Comandă. Modifică numerele de ordine ale liniilor din programul curent, începînd cu linia de număr ⟨număr linie vechi⟩ (expresie numerică pozitivă, cu valoarea implicită a primei linii din program) folosind în acest scop o secvență de renumerotare ce începe cu ⟨număr linie nou⟩ (expresie numerică pozitivă, cu valoare implicită 10) și are un pas ⟨increment⟩ (expresie numerică pozitivă cu valoare implicită 10).

Observație. Interpretorul GW–BASIC asigură actualizarea automată și a referințelor la liniile renumerotate gen **ELSE, ERL, GOTO, GOSUB, THEN, ON . . . GOTO, ON . . . GOSUB, RESTORE, RESUME.**

RESET (C)

RESET

Comandă. Toate fișierele de date din program care sînt deschise vor fi închise, oricare ar fi dispozitivele pe care sînt montate. În cazul fișierelor de ieșire disc se asigură și scrierea tuturor blocurilor rămase în memorie.

RESTORE(S)

RESTORE [⟨număr linie⟩]
10 **READ** A, B, C
20 **RESTORE**
30 **READ** D, E, F
40 **DATA** 11., 17., 14. 'SE VOR CITI ACELEAȘI DATE CA ÎN A, B, C

Enunț. Se (re)stabilește poziția curentă într-un bloc de date (definit de enunțuri de tip **DATA** și accesibil prin enunțuri de tip **READ**) la subblocul ⟨număr linie⟩ (implicit primul bloc **DATA**).

Cuvinte cheie asociate. **DATA, READ**

RESUME(S)

RESUME ⟨punct reluare⟩
cu
⟨punct reluare⟩ : :=0|**NEXT**|⟨număr linie⟩

10 **ON ERROR GOTO** 1000
.
.
.
1000 **IF** (**ERR**=230) **AND** (**ERL**=90) **THEN PRINT** "MAI ÎNCERCAȚI"
1010 **RESUME** 70

Enunţ. Se autorizează reluarea execuţiei programului GW-BASIC după ce a avut loc tratarea printr-o rutină utilizator a unei întreruperi provocate de un eveniment semnificativ (sosirea de caractere într-un tampon de comunicaţii; detectarea unei erori GW–BASIC; acţionarea unei chei speciale; atingerea unei limite inferioare a capacităţii tamponului muzicii de fond etc.), reluarea efectuîndu-se cu una din opţiunile ⟨punct reluare⟩:

– 0 (implicit): de la enunţul ce a provocat evenimentul (al cărui context de execuţie a fost eventual modificat în ritm de tratare);

– **NEXT** de la enunţul imediat următor enunţului ce a provocat evenimentul;

– ⟨număr linie⟩: de la enunţul cu numărul de ordine specificat.

RIGHT$(F)

RIGHT$ (⟨şir⟩, ⟨lungime⟩)
10 X$="PROTOPOPESCU"
20 **PRINT RIGHT$** (X$, 7)
RUN
POPESCU

Funcţie. Furnizează un subşir sufix al unui şir reprezentînd valoarea expresiei şir ⟨şir⟩ şi avînd un număr de caractere egal cu ⟨lungime⟩ (expresie numerică întreagă cu valori 0–255). O valoare 0 pentru ⟨lungime⟩ semnifică şirul **NULL**, iar o valoare superioară lungimii şirului iniţial conduce la restituirea integrală a acestuia.

Cuvinte cheie asociate. **LEFT$, LEN, MID$**

RMDIR(C)

RMDIR ⟨traseu⟩
RMDIR "CATAL 11" 'DESFIINŢEAZĂ SUBCATALOGUL CATAL 11 AL CATALOGULUI CURENT

Comandă. Este desfiinţat (sub)catalogul terminal din şirul ⟨traseu⟩, cu condiţia ca acesta să fie vid.

Cuvinte cheie asociate. **CMDIR, MKDIR**

RSET(S)

RSET ⟨var şir⟩=⟨expresie şir⟩
150 X$=**SPACES** (30)
160 **RSET** Y$=X$

Enunţ. Valoarea ⟨expresie şir⟩ este atribuită, aliniată la dreapta, unei variabile şir (⟨var şir⟩ de sine stătătoare, sau care a fost definită de enunţul **FIELD** într-o zonă tampon pentru acces aleator.

Cuvinte cheie asociate. **FIELD, LSET, MKD$, MKI$, MKS$**

RND(F)

RND [(⟨expresie numerică⟩)]
10 **FOR** I=1 **TO** 5
20 **PRINT INT** (**RND** * 1000);
30 **NEXT**

RUN
120 650 860 720 790
Ok

Funcţie. Se calculează un număr pseudoaleator uniform distribuit în intervalul 0–1, ţinînd cont de 'sămînţa' specificată implicit (sau explicit printr-un enunţ **RANDOMIZE** anterior) şi potrivit caracteristicilor opţiunii ⟨expresie numerică⟩:

– pozitivă (implicit): se furnizează următorul număr din şir
– negativă: se resetează generarea pornind de la 'sămînţă' şi furnizîndu-se primul număr din şir
– nulă: se furnizează numărul curent din şir (de la ultima invocare a funcţiei).

Cuvinte cheie asociate: **RANDOMIZE**

RUN(C)

RUN [⟨program⟩]
cu
⟨program⟩ : : = ⟨număr linie⟩ | ⟨specificator fişier⟩ [,R]
RUN 200 'SE COMANDĂ EXECUŢIA PROGRAMULUI CURENT DE LA LINIA 200
. . . .
RUN "A : \CATAL1\CATAL11\PROGR1" 'EXECUŢIE PROGR1

Comandă. Provoacă execuţia programului curent din memorie începînd cu linia de rang ⟨număr linie⟩ (expresie numerică pozitivă, cu valoarea implicită rangul primei linii executabile din programul curent), sau încărcarea în memorie a unui program GW–BASIC reţinut pe disc sub identificarea completă ⟨specificator fişier⟩ (prin mijlocirea unui enunţ **SAVE** anterior), păstrînd eventual deschise fişierele de date deschise în momentul încărcării noului program (opţiunea **R**).

Extensia explicită a fişierului disc este **.BAS.**

Cuvinte cheie asociate. **SAVE**

SAVE(C)

SAVE ⟨specificator fişier⟩ [⟨opţiuni salvare⟩]
cu
⟨opţiuni salvare⟩ : : =**A** | **P**
SAVE "PROG2.X" 'Salvare program curent în (sub)catalogul curent sub numele PROG2.X
SAVE "P57" 'Salvare program curent în (sub)catalogul curent sub numele P57 . BAS

Comandă. Programul curent este salvat pe disc sub forma unui fişier reperat în sens MS . DOS cu ⟨specificator fişier⟩. Salvarea se face de regulă în format binar condensat, exceptînd utilizarea opţiunii **A** care-l salvează ca fişier ASCII.

Cu opţiunea **P** se efectuează salvarea binară codificată intern, introducîndu-se o protecţie ce interzice listări (enunţ **LIST**) sau editări (enunţ **EDIT**) ulterioare.

Numele de extensie explicit este **. BAS.**

Cuvinte cheie asociate. **LOAD RUN**

SCREEN(F)

SCREEN (⟨rînd⟩, ⟨coloană⟩ [,⟨condiţie⟩])
110 **REM** DACĂ ÎN LINIA 10, COLOANA 25 ESTE UN 'B'
SE AFIŞEAZĂ 66
120 **PRINT** "CARACTERUL ESTE : " ; **SCREEN** (10, 25);
130 **REM** DACĂ CULOAREA ÎN LINIA 10, COLOANA 25
ESTE DE COD 3
140 **PRINT** "CODUL CULORII ESTE : " ; **SCREEN** (10, 25, 1)

Funcţie. Pentru caracterul de pe linia ⟨rînd⟩ (expresie numerică cu valori 1–25) şi coloana ⟨coloană⟩ (expresie numerică cu valori 1–40 sau 1–80) se furnizează o caracteristică ce depinde de valoarea ⟨condiţie⟩ (expresie numerică validă, sau mai general logică) după cum urmează:

– valoare nenulă ('adevărat'): se furnizează numărul culorii (0–3) caracterului afişat
– valoare nulă ('fals'): se furnizează codul ASCII (0–255) al caracterului afişat, E opţiunea implicită.

SCREEN(S)

SCREEN [⟨mod⟩] [,[⟨aparenţă⟩]; [⟨actpag⟩] [,⟨vizpag⟩]]]
10 **SCREEN** 0, 1, 0, 0 'Mod text color, pagini active şi vizuale 0
20 **SCREEN** , , 1, 2 'Se trece la pagina activă 1 şi de vizualizare 2
30 **SCREEN** 2 'Se comută în mod grafic de rezoluţie mare
40 **SCREEN** 1, 1 'Se comută în rezoluţie medie, color
50 **SCREEN** , 0 'Se comută în alb negru, cu aceeaşi rezoluţie

Enunţ. Se stabilesc caracteristicile ecranului de afişat după cum urmează:

a) Rezoluţia prin ⟨mod⟩ (expresie numerică cu valori 0–255)
0 – mod text (25 linii×80 coloane)
1 – rezoluţie medie (320×200 elemente imagine)
2 – rezoluţie mare (640×200 elemente imagine)
⩾3 – rezoluţie ultra înaltă (640×400 elemente de imagine)

b) Aparenţa prin ⟨aparenţa⟩ (expresie întreagă cu valori 0–1) avînd semnificaţia:

	⟨mod⟩=0	⟨mod⟩=1	alte ⟨mod⟩
0	alb-negru	color	ignorat
1	color	alb-negru	

c) În ⟨mod⟩=0 (adică text) se poate selecta o pagină activă ⟨actpag⟩ (expresie întreagă cu valori 0–7 sau 0–3, funcţie de lăţime ecran, implicit 0) şi o pagină vizuală ⟨vizpag⟩ (expresie întreagă cu valori 0–7 sau 0–3, funcţie de lăţime ecran, implicit valoarea ⟨actpag⟩)

Observații

● Teoretic lățimea ecranului se stabilește cu enunțul **WIDTH.** Totuși există următoarele efecte ale enunțului **SCREEN** provocate de ⟨mod⟩:

0, 2, 3 – 80 coloane
1 – 40 coloane

● În absența oricărui enunț **SCREEN,** efectul e același ca al enunțului:
SCREEN 0, 0, 0, 0, ' Mod text, alb-negru

Cuvinte cheie asociate. **WIDTH**

SGN(F)

SGN (⟨expresie numerică⟩)

```
145 REM FUNCȚIE DE SEMNUL ARGUMENTULUI SE EXECUTĂ PROCEDURILE DE
    LA LINIILE 300, 400, 500
150 ON SGN (X)+2 GOTO 300, 400, 500
```

Funcție. Se restituie valorile 1, 0, –1 după cum valoarea argumentului (expresie numerică) este respectiv strict pozitivă, nulă sau negativă.

SHELL(C)

SHELL [⟨expresie șir⟩]

```
.
.
.
530 OPEN "FISN.DAT" FOR OUTPUT AS #1 ' DESCHIDERE FIȘIER NESORTAT
.
.
.
620 CLOSE #1 ' INTERVALUL 530–620 S-AU SCRIS DATE ÎN FIȘIER
630 REM SE APELEAZĂ VIA MS–DOS PROGRAMUL DE SORTARE
640 REM CU REDIRECTARE STDIN LA FISN. DAT ȘI STDOUT LA FISSORT. DAT
650 SHELL "SORT <FISN. DAT> FISSORT . DAT"
660 REM S-A TERMINAT SORTAREA, GW–BASIC ÎȘI REIA ACTIVITATEA
670 OPEN "FISSORT.DAT" FOR INPUT AS #1 ' SE PRELUCREAZĂ
    FIȘIERUL SORTAT
.
.
.
950 CLOSE #1 ' ÎN INTERVALUL 670–950 S-AU EFECTUAT PRELUCRĂRI PE FIȘIER
```

Comandă: Se creează un proces fiu (în sens MS-DOS) prin încărcarea unei copii secundare a procesorului de comenzi MS–DOS, COMMAND. COM, căruia eventual i se transmite linia de comandă ⟨expresie șir⟩ spre prelucrare. În lipsa argumentului conversația cu procesorul secundar decurge de la consolă, pînă ce se dă comanda MS–DOS **EXIT.**

Pe durata derulării procesului fiu, interpretorul GW–BASIC rămîne rezident.

SIN(F)

SIN (⟨expresie numerică⟩)

```
PRINT SIN (1.5)
 .9974951
```

SWAP(S)

```
SWAP ⟨variabilă 1⟩, ⟨variabilă 2⟩
10 I1=17 : X$="IANUARIE" : I2=1943
20 PRINT I1; X$; I2
30 SWAP I1, I2
40 PRINT I1; X$; I2
RUN
   17 IANUARIE 1943
   1943 IANUARIE 17
```

Enunţ. ⟨variabilă 1⟩ primeşte valoarea ⟨variabilă 2⟩ şi invers. Variabilele trebuie să fie de acelaşi tip.

SYSTEM(C)

SYSTEM

Comandă. Toate fişierele deschise sînt automat închise, interpretorul GW–BASIC îşi încheie activitatea, restituind controlul sistemului de operare (MS–DOS). Dacă interpretorul fusese lansat dintr-un fişier de comenzi, controlul este restituit procesorului fişierului de comenzi.

TAB(F)

```
TAB ⟨tabulator⟩
10 X$="DUMITRAŞCU"; Y$="LIVIU"; V%=40
20 PRINT X$ TAB (30) Y$=TAB (50) V%
```

Funcţie. Cursorul ecranului (mod text) sau unul de scriere (imprimantă) sînt poziţionate în coloana dată de modulo lungimea liniei relative la ⟨tabulator⟩ (expresie întreagă cu valori 1–255, dar fără a depăşi lungimea liniei minus 1).

Observaţie. Implicit un caracter ' ; ' este presupus că urmează funcţiei **TAB** într-o listă de expresii de ieşire, specificată în enunţurile **LPRINT, PRINT** sau **PRINT #**.

TAN(F)

```
TAN (⟨expresie numerică⟩)
X=.785495
PRINT TAN (X)
1.0000
```

Funcţie. Calculează tangenta argumentului ⟨expresie numerică⟩.
Cuvinte cheie asociate. **SIN, COS**

TIME$(S)

```
TIME$=⟨expresie şir⟩
TIME$="07 : 53 : 07"
```

Enunţ. Se stabileşte orologiul curent la valoarea ⟨expresie şir⟩ care poate fi de una din valorile "hh", "hh : mm", "hh : mm : ss", în grupajele care exprimă ora (hh), minutul (mm), secunda (ss) putînd fi suprimate zerourile iniţiale.

Orizontul de timp acceptat este de 24 ore.

Cuvinte cheie asociate. **DATE$**

TIME$ (F)

⟨var şir⟩=**TIME$**
TIME$="07 : 53 : 57"
PRINT TIME$
07 : 54 : 01

Funcţie. Valoarea tip constantă şir a orologiului curent de forma "hh : mm : ss" atribuită variabilei şir ⟨varşir⟩.

De asemenea, această valoare se atribuie şi unei variabile 'ascunse' dacă funcţia este citată într-o expresie tip şir.

Cuvinte cheie asociate. **DATE$**

TIMER (F)

TIMER
.
.
.
100 **REM** SE VA MARCA DURATA SECVENŢEI ÎNTRE LINIILE 110 şi 210
110 T1=**TIMER**
.
.
210 T2=**TIMER**
220 **PRINT** "DURATA SECVENŢEI 110 . . . 210 ÎN SECUNDE
A FOST: " ; T2–T1

Funcţie. Furnizează un număr real în simplă precizie ce arată numărul de secunde scurse de la reiniţializarea sistemului de calcul, sau de la miezul nopţii.

TIMER OFF (S)

TIMER OFF

Enunţ. Dezactivează (inhibă) întreruperea provocată de evenimentul ce constă din scurgerea unui interval de timp (şi care este în atenţia unui enunţ **ON TIMER** (n) **GOSUB**).

Cuvinte cheie asociate. **TIMER ON, TIMER STOP**

TIMER ON (S)

TIMER ON

Enunţ: Activează întreruperea provocată de evenimentul ce constă din scurgerea unui interval de timp (şi care se află în atenţia unui enunţ **ON TIMER** (n) **GOSUB**).

Cuvinte cheie asociate. **TIMER OFF, TIMER STOP**

TIMER STOP (S)

TIMER STOP

Enunţ. Suspendă luarea în considerare (printr-o rutină tip **ON TIMER** (n) **GOSUB**) a evenimentului provocat de scurgerea unui interval de timp, amînînd această operaţie pînă la o activare explicită (cu un enunţ **TIMER ON**).

Cuvinte cheie asociate. **TIMER OFF, TIMER ON**

Se recomandă utilizarea funcţiei pentru executarea subşirurilor din enunţurile **DRAW** şi **PLAY** din programe GW-BASIC care vor fi compilate mai tîrziu.

VIEW (SGR)

VIEW [[**SCREEN**][(⟨vx1⟩, ⟨vy1⟩)–(⟨vx2⟩, ⟨vy2⟩)
[,⟨culoare⟩[,⟨frontieră⟩]]]]

```
110
120 CLS
130 VIEW 'VIZORUL PRIM OCUPĂ TOT ECRANUL
140 VIEW (10, 10)–(300, 180),, 1 'AL DOILEA VIZOR
140 CLS
150 LINE (0, 0)–(310, 190), 1
160 LOCATE 1, 11 : PRINT "VIZORUL DOI"
170 VIEW SCREEN (50, 50)–(250, 150),, 1 'AL TREILEA VIZOR
180 CLS 'SE ŞTERGE NUMAI ÎN VIZOR 3
190 LINE (300, 0)–(0, 199), 1
200 LOCATE 9, 9 : PRINT "VIZORUL TREI"
210 VIEW SCREEN (80, 80)–(200, 125),, 1 'CEL MAI INTERIOR VIZOR
220 CLS
230 CIRCLE (150, 100), 20, 1
240 LOCATE 11, 9 : PRINT "VIZORUL PATRU"
```

Enunţ. Defineşte un vizor reprezentat de limitele fizice ale suprafeţei de afişare (ecranului) în care se va face proiectarea imaginilor grafice, limite reprezentate de abscisa ⟨vx1⟩ şi ordonata ⟨vy1⟩ a colţului NV şi abscisa ⟨vx2⟩ şi ordonata ⟨vy2⟩ a colţului SE (simbolurile metalingvistice sînt expresii numerice întregi). Opţional, vizorul poate fi umplut cu un cod de ⟨culoare⟩ (expresie numerică cu valori 0–3) şi eventual cu trasarea unei frontiere ce delimitează vizorul (dacă se specifică o valoare nenulă pentru expresie numerică ⟨frontieră⟩).

Opţiunea **SCREEN** arată că toate coordonatele grafice se vor calcula relativ la întregul ecran (deci absolut) în caz contrar sînt relative la vizor.

Cuvinte cheie asociate. **VIEW PRINT**, **WINDOW**

VIEW PRINT (S)

VIEW PRINT [⟨linie 1⟩ **TO** ⟨linie 2⟩]

```
10 REM SE CREEAZĂ O FEREASTRĂ DE 6 LINII LA ÎNCEPUTUL ECRA-
   NULUI
20 VIEW PRINT 1 TO 5
```

Enunţ. Se fixează limitele unei ferestre de text stabilite între numerele de linie ⟨linie 1⟩ şi ⟨linie 2⟩ (expresii numerice întregi cu valori 1–24).

Toate funcţiile de scriere, poziţionare, editare, defilare au loc în această fereastră.

WAIT (S)

WAIT ⟨poartă⟩, ⟨i⟩[,⟨j⟩]

```
100 WAIT 32, 2
```

Enunţ. Se suspendă execuţia programului pînă ce un canal de comunicaţie maşină ⟨poartă⟩ (expresie întreagă cu valori 0–65535) elaborează

un anume şablon de biţi. Mai precis, datele citite de la ⟨poartă⟩ sînt puse în operaţie **SAU EXCLUSIV** cu expresia întreagă (cu valori: 0–255) ⟨j⟩ şi rezultatul pus în operaţie logică şi cu expresia întreagă (valori 0–255) ⟨i⟩. Dacă rezultatul este nul, atunci aşteptarea durează, reiterîndu-se citirea de la canalul de comunicaţii.

WHILE . . . WEND(S)

WHILE ⟨condiţie⟩
.
.
.
.
WEND

```
10 REM SE FACE SUMA PRIMILOR ZECE ÎNTREGI
20 M%=0 : I%=1
30 WHILE I%<11
40    M%=M%+I% : I%=I%+1
50 WEND
```

Enunţ. Se execută corpul ciclului delimitat de cuvintele cheie **WHILE** şi **WEND** de un număr repetat de ori, cît timp valoarea expresiei logice ⟨condiţie⟩ este adevărată. Dacă prin ⟨condiţie⟩ se furnizează o expresie numerică, atunci valoarea nenulă se interpretează ca 'adevărat' iar o valoare nulă ca 'fals'.

WIDTH LPRINT(S)

WIDTH LPRINT ⟨dimensiune⟩

Enunţ. Stabileşte lungimea liniei de imprimantă la ⟨dimensiune⟩ (expresie întreagă cu valori 0–255).

WIDTH(S)

WIDTH ⟨obiect⟩, ⟨dimensiune⟩
cu:
⟨obiect⟩ : : =⟨dispozitiv⟩ | ⟨număr canal⟩
⟨dispozitiv⟩ : : ="SCRN : " | "LPT1 : " | "LPT2 : " | "LPT3 : " | "COM1 : " | "COM2 : " | "COM3 : " | "COM4 : "

```
10 PRINT "1234567890"
RUN
1234567890
WIDTH 3
RUN
123
456
789
0
```

Enunţ. Se fixează la ⟨dimensiune⟩ (expresie întreagă cu valori 0–255) lungimea liniei pentru un ⟨dispozitiv⟩ (constantă şir desemnînd un dispozitiv MS-DOS) înainte de a fi deschis sau pentru un fişier căruia i se va atribui la deschiderea ulterioară numărul GW-BASIC ⟨număr canal⟩.

WINDOW(SGR)

WINDOW [[**SCREEN**](⟨wx1⟩, ⟨wy1⟩)–(⟨wx2⟩, ⟨wy2⟩)]
100 **WINDOW** (–10, –20), (30, 40)
120 **VIEW** (40, 60)–(100. 120)
130 **LINE** (–5, –12)–(25, 37)

Enunţ. Defineşte dimensiunile logice (în coordonate univers) ale vizorului curent (o fereastră de fapt) prin abscisa ⟨wx1⟩ şi ordonata ⟨wy1⟩ a colţului SV şi abscisa ⟨wx2⟩ şi ordonata ⟨wy2⟩ a colţului NE (simbolurile metalingvistice sînt expresii numerice. Cu opţiunea **SCREEN** are loc intervertirea axelor X şi Y. Automat spaţiul logic definit de **WINDOW** este proiectat în spaţiul fizic definit de enunţul **VIEW** (implicit tot ecranul).

Cuvinte cheie asociate. **VIEW**

WRITE(S)

WRITE [⟨lista de expresii⟩]
cu ⟨listă de expresii⟩ : : =⟨expresie⟩ | ⟨lista de expresii⟩, ⟨expresie⟩
10 A=80 : B=90 : C$="GATA"
20 **WRITE** A, B, C$
RUN
80, 90, "GATA"

Enunţ. Se scriu pe ecran valorile ⟨expresie⟩ utilizînd separatorul ',', încadrarea valorilor tip şir între ghilimele, iar în final se inserează automat secvenţa ⟨CR⟩, ⟨LF⟩.

Cuvinte cheie asociate. **READ**, **WRITE#**

WRITE#(S)

WRITE# ⟨număr canal⟩, ⟨listă de expresii⟩
10 X$="BASIC"
20 Y$="45 ANI"
30 **WRITE#**1, X$, Y$ 'PE DISC APARE : "BASIC", "45 ANI"

Enunţ. Valorile din ⟨listă de expresii⟩, separate de caracterul ', ' (iar valorile şir încadrate între ghilimele) sînt înscrise în fişierul căruia i s-a asociat la deschidere numărul GW-BASIC ⟨număr canal⟩.

Cuvinte cheie asociate. **INPUT#**, **READ**, **WRITE**

SINTEZA 18

Proiectarea asistată de calculator şi reprezentări geometrice în BASIC

☐ Elemente de proiectare asistată de calculator (PAC)

Caracteristici ale procesului de proiectare

În zilele noastre, ca urmare a progresului ştiinţific şi tehnic societatea umană devine tot mai mult şi o **societate informatizată,** caracterizare ce include, printre altele şi pătrunderea utilizării sistemelor automate de stocare, manipulare şi prelucrare a informaţiilor în cele mai variate domenii de activitate. Desigur în aceste domenii, acela al **proiectării** ocupă un loc de frunte. Ansamblul acţiunilor umane care constituie patrimoniul activităţii de proiectare vizează întotdeauna un produs nou, indiferent de natura acestuia, avînd drept ţel transformarea unei teme în documentaţia de realizare a produsului. Proiectul reprezintă din punct de vedere semantic o "lucrare tehnică întocmită pe baza unei **teme date,** care cuprinde **calcule tehnico-economice, desene, instrucţiuni** necesare executării unei construcţii, unei maşini etc." [54].

Elementul esenţial care trebuie reţinut în conexiune cu activitatea de proiectare este acela că nici măcar la încheierea acesteia produsul nou **nu există ca atare,** el avînd doar o prezenţă **virtuală** sub forma unui **model**; tocmai descrierea convenţională a acestui model este reprezentată în cadrul documentaţiei de realizare a produsului.

În timpul activităţilor de proiectare **tema iniţială** care include caracteristicile pe care trebuie să le întrunească viitorul produs şi motivaţia unei astfel de solicitări este transformată într-o **documentaţie** care cuprinde printre altele: structura detaliată a produsului; proprietăţile fiecărei componente; raporturile între componente; comportamentul estimat al viitorului produs în mediul său probabil de evoluţie; resursele necesare realizării produsului; resursele necesare funcţionării produsului la parametrii estimaţi; eficienţa economică a funcţionării produsului.

Caracterul de proces al activităţii de proiectare

Pe durata elaborării unui proiect se pot identifica o serie de operaţii, stări, fenomene prin care se efectuează diversele lucrări, se produc transformările necesare, ceea ce conferă un caracter de **proces** activităţii de proiectare. Procesele sînt create şi controlate pe parcursul existenţei lor de

către componente speciale: colecţia de informaţii accesibilă procesului; "adresa" următoarei operaţii (instrucţiuni) de executat; ansamblul parametrilor de stare ai procesului; colecţia de proceduri pentru tratarea evenimentelor previzibile, susceptibile de a întrerupe desfăşurarea normală a procesului; structura informaţională utilizată pentru salvarea stării curente a procesului (contextul său cu alte cuvinte) în momentul apariţiei unei întreruperi şi pentru restaurarea contextului general în momentul rezolvării situaţiei de întrerupere.

În literatura de specialitate se prezintă diverse modele ale procesului de proiectare, modele independente de tipul şi caracteristicile obiectului ce se proiectează. Un asemenea model este prezentat în fig. 18.1.

Procesul de proiectare nu este izolat, de sine stătător, ci legat şi de alte procese:

a) procesul de nivel superior care îl creează în virtutea îndeplinirii unei sarcini precise şi căruia îi raportează rezultatele;

b) procesul asociat mediului în care se desfăşoară ansamblul activităţilor de proiectare şi avînd de îndeplinit sarcini specifice: recepţia cereri-

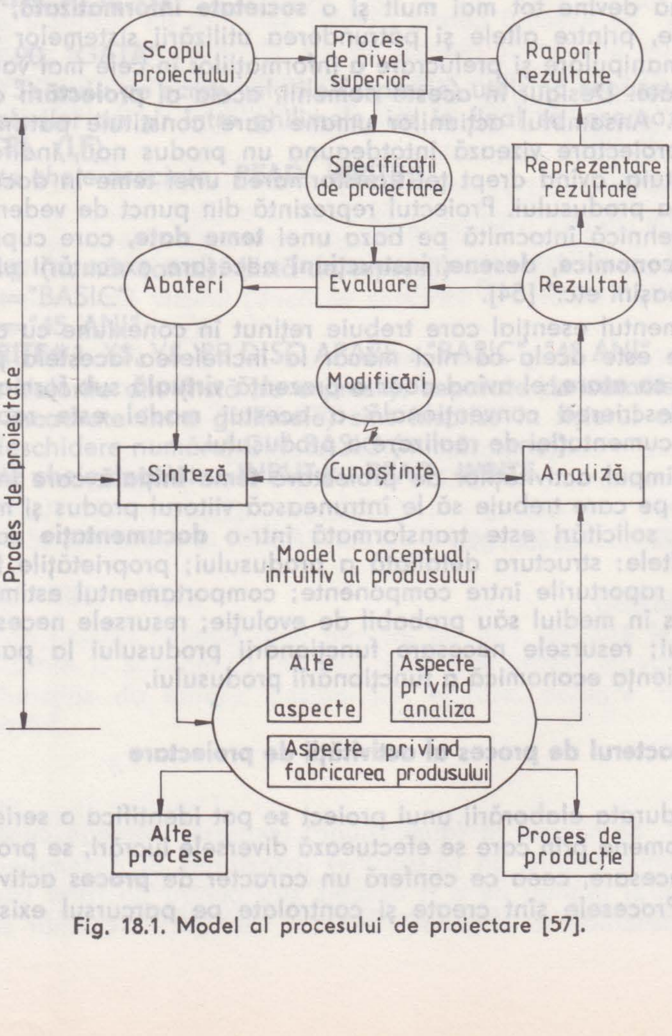


Fig. 18.1. Model al procesului de proiectare [57].

lor privind crearea unor noi (sub)teme de proiectare; definirea unor metode de reprezentare a specificaţiilor de proiectare şi a rezultatelor, precum şi metode de creare, conducere şi finalizare a noilor (sub)teme de proiectare; conducerea şi coordonarea propriu-zisă a noilor (sub)teme de proiectare; gestionarea resurselor folosite în mod curent de diversele subprocese de proiectare găzduite de mediul considerat; rezolvarea conflictelor de acaparare a resurselor, urmărindu-se în acelaşi timp o optimizare a utilizării acestora; conducerea achiziţionării, stocării şi reprezentării cunoştinţelor.

Modelul din fig. 18.2 evidenţiază şi **caracterul iterativ** al procesului de proiectare.

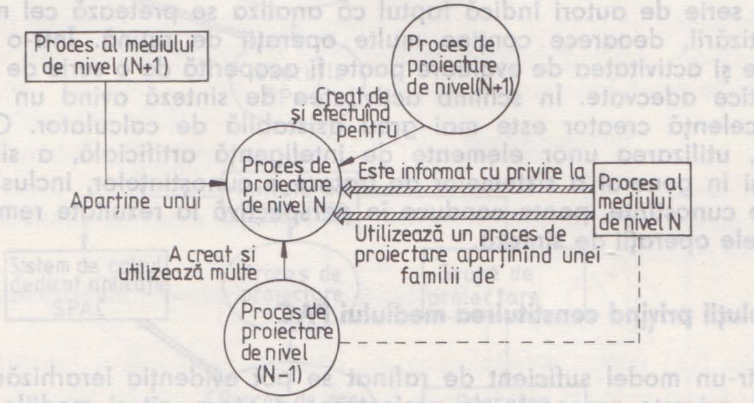


Fig. 18.2. Bloc de component elementar al unui proces de proiectare [57].

Iniţial scopul şi obiectivele proiectării sînt prezentate unui proces de nivel superior care întocmeşte o temă de proiectare (sub forma unor specificaţii) care reprezintă sarcinile de proiectare pentru un (sub)proces de proiectare de nivel imediat inferior.

a) Una din activităţile majore ale acestui (sub)proces este aceea de **sinteză** prin care se încearcă obţinerea unei soluţii la tema dată. Soluţia are ca suport un ansamblu de cunoştinţe, dobîndite pe diverse căi (experienţa trecută, documentare şi studiu, autoinstruire etc.) şi este reprezentată de o **schemă conceptuală** de fapt un **model** al viitorului produs (conţinînd aspecte structurale, funcţionale, grafice etc.).

b) O altă activitate majoră a (sub)procesului de proiectare este aceea de **analiză** prin care modelul este examinat după o serie de criterii bine determinate (stipulate în metodologia de proiectare) efectuîndu-se o serie de verificări de completitudine şi consistenţă, materializate prin aşa-numitele **rezultate.**

c) Activitatea de **evaluare** confruntă rezultatele obţinute în faza de analiză cu specificaţiile iniţiale, orice **abatere** reiterînd ciclul sinteză-analiză-evaluare pînă ce acestea se înscriu în limitele toleranţelor admisibile (ideal se anulează).

d) Periodic, rezultatele presupuse acceptabile din punctul de vedere al (sub)procesului de proiectare iniţiat se prezintă sub formă de **realizări** (printr-o transformare specială definită în activitatea de **prezentare** a rezultatelor) procesului de nivel superior. Acesta, din confruntarea cu scopurile iniţiale, poate valida soluţia sau modifica, detalia, clarifica specifica-

ţiile de proiectare, reiniţiind (sub)procesul de proiectare chemat a le traduce în viaţă.

O problemă legitimă care se pune, este aceea a posibilităţii de informatizare a principalelor categorii de activităţi ale unui proces de proiectare:

– sinteza; analiza; evaluarea; transformările de reprezentări.

O primă condiţie pentru automatizarea unei succesiuni de operaţii este aceea a formalizării descrierii acestor operaţii, a obiectelor cărora li se aplică şi a înlănţuirii şi intercondiţionării lor.

O serie de autori indică faptul că analiza se pretează cel mai bine informatizării, deoarece conţine multe operaţii de rutină. Într-o măsură oarecare şi activitatea de evaluare poate fi acoperită de o serie de produse informatice adecvate. În schimb activitatea de sinteză avînd un caracter prin excelenţă creator este mai greu asistabilă de calculator. Cu toate acestea, utilizarea unor elemente de inteligenţă artificială, a sistemelor expert şi în general a sistemelor de tratare a cunoştinţelor, inclusiv a bazelor de cunoştinţe, poate conduce în perspectivă la rezultate remarcabile şi în unele operaţii de sinteză.

Soluţii privind constituirea mediului PAC

Într-un model suficient de rafinat se pot evidenţia ierarhizări atît în ceea ce priveşte procesele de proiectare ca atare, cît şi mediile în care aceste procese evoluează. O astfel de situaţie este ilustrată în fig. 18.2.

Pornind de la o asemenea ierarhizare partea efectiv informatizată a procesului de proiectare poate fi privită ca un subproces de proiectare asistată de calculator, care evoluează într-un mediu specific oferit, pe diverse niveluri de detaliere de cadrul organizaţional suport al echipamentelor de prelucrare automată a datelor, de înseşi aceste echipamente, sistemele de operare şi produsele informatice care efectuează sarcinile specifice. În figura 18.3(a, b) se arată diverse variante.

a) Soluţia din fig. 18.3(a) reprezintă abordarea tradiţională, care limitează de fapt caracterul interactiv conţinut în intimitatea noţiunii de proiectare asistată de calculator. Ea rămîne totuşi aplicabilă într-o serie de cazuri cum ar fi:

– unitatea de proiectare nu dispune de forţă de calcul proprie, lucrările proprii de informatică rulînd în concurenţă cu alte produse informatice pe un calculator central (situat eventual la distanţă);

– performanţele cerute programelor sînt foarte mari, atît în ceea ce priveşte volumul memoriei interne folosite şi a memoriei externe necesare, cît şi rapiditatea răspunsului, deci viteza de prelucrare.

b) Abordarea din fig. 18.3(b) se referă la situaţia în care unitatea de proiectare (reprezentată în mod ideal de o grupă de proiectare sau un atelier ori birou de proiectare) dispune de o capacitate de calcul proprie de tipul unui minisistem, microsistem, sau în ultimii ani chiar un calculator profesional ori personal. De notat că actualmente performanţele acestor capacităţi de calcul dedicate sînt remarcabile, ele fiind dotate şi cu sisteme de operare performante care dispun de o serie de facilităţi (multiprogra-

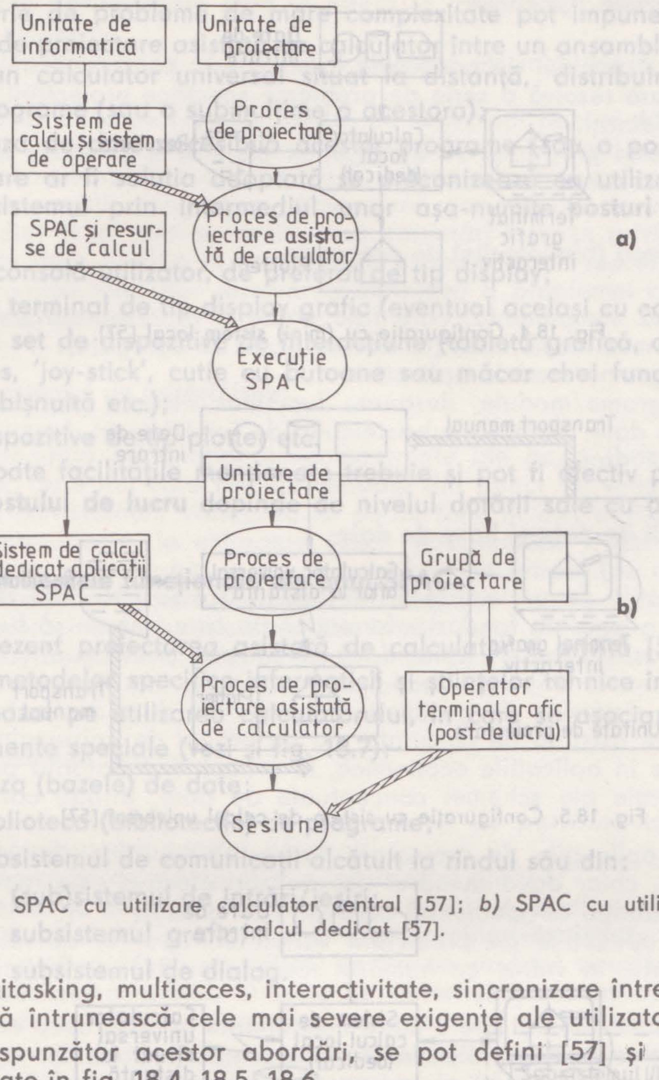


Fig. 18.3. *a)* SPAC cu utilizare calculator central [57]; *b)* SPAC cu utilizare sistem de calcul dedicat [57].

mare, multitasking, multiacces, interactivitate, sincronizare între lucrări etc.) capabile să întrunească cele mai severe exigenţe ale utilizatorilor.

Corespunzător acestor abordări, se pot defini [57] şi unele soluţii practice, date în fig. 18.4, 18.5, 18.6.

Soluţia prezentată în fig. 18.4 este la ora actuală cea mai convenabilă din punctul de vedere al cerinţelor unei echipe de proiectanţi. (Faptul că ea sugerează un domeniu de aplicaţii din ramura proiectării constructiv-tehnologice nu este esenţial).

În varianta din fig. 18.5 deşi se poate păstra caracterul interactiv al procesului de proiectare asistată prin conexiune pe linie de comunicaţii seturile de date de intrare şi cele de ieşire sînt transportate la acesta şi aduse de la acesta prin mijloace manuale.

Soluţia ilustrată de fig. 18.6 răspunde unor situaţii speciale legate de depăşirea posibilităţilor hardware şi software ale sistemului de calcul dedicat, cînd trebuie să se facă apel "on-line" la o putere de calcul superioară. În trecut, un exemplu clasic din această categorie era pferit de

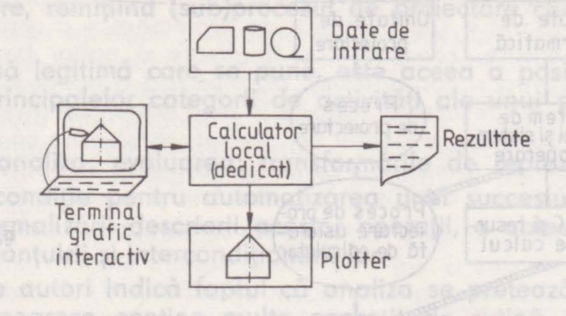


Fig. 18.4. Configuraţie cu (mini) sistem local [57].

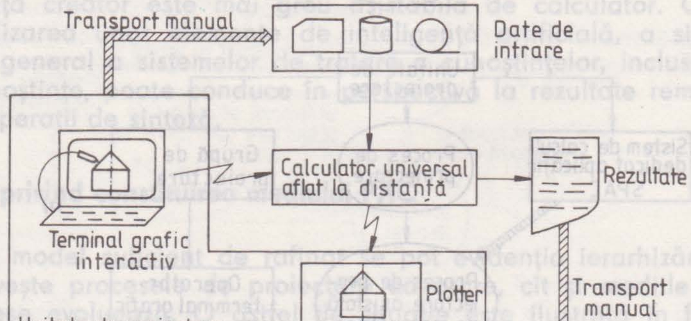


Fig. 18.5. Configuraţie cu sistem de calcul universal [57].

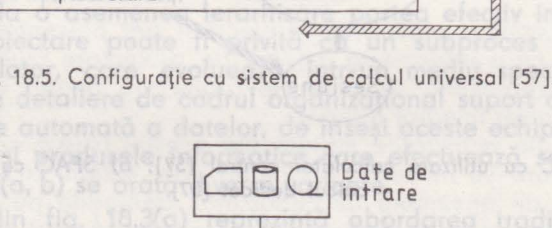


Fig. 18.6. Configuraţie cu sistem local conectat cu calculator universal situat la distanţă [57]

aplicaţii numerice vizînd aplicarea metodei elementului finit. În prezent însă s-au realizat pe plan mondial pachete de programe deosebit de performante utilizînd metoda elementului finit, dar putînd rula pe calculatoare personale.

O serie de probleme de mare complexitate pot impune distribuirea sistemelor de proiectare asistată de calculator între un ansamblu de sisteme locale şi un calculator universal situat la distanţă, distribuindu-se [57]:

– programe (sau o submulţime a acestora);

– baza de date accesibilă acestor programe (sau o porţiune a sa).

Oricare ar fi soluţia adoptată se preconizează ca utilizatorul să interfaţeze sistemul prin intermediul unor aşa-numite **posturi de lucru** incluzînd:

– o consolă utilizator, de preferat de tip display;

– un terminal de tip display grafic (eventual acelaşi cu consola);

– un set de dispozitive de interacţiune (tabletă grafică, digitizor, creion luminos, 'joy-stick', cutie cu butoane sau măcar chei funcţionale la o tastatură obişnuită etc.);

– dispozitive de tip plotter etc.

Nu toate facilităţile menţionate trebuie şi pot fi efectiv prezente, dar puterea **postului de lucru** depinde de nivelul dotării sale cu acestea.

Componentele funcţionale ale unui SPAC

În prezent proiectarea asistată de calculator e privită [57] ca o integrare a metodelor specifice informaticii şi ştiinţelor tehnice într-un sistem complex, bazat pe utilizarea calculatorului, în care se asociază şi o serie de componente speciale (vezi şi fig. 18.7):

– baza (bazele) de date;

– biblioteca (bibliotecile) de programe;

– subsistemul de comunicaţii alcătuit la rîndul său din:

- (sub)sistemul de intrări/ieşiri;
- subsistemul grafic;
- subsistemul de dialog.

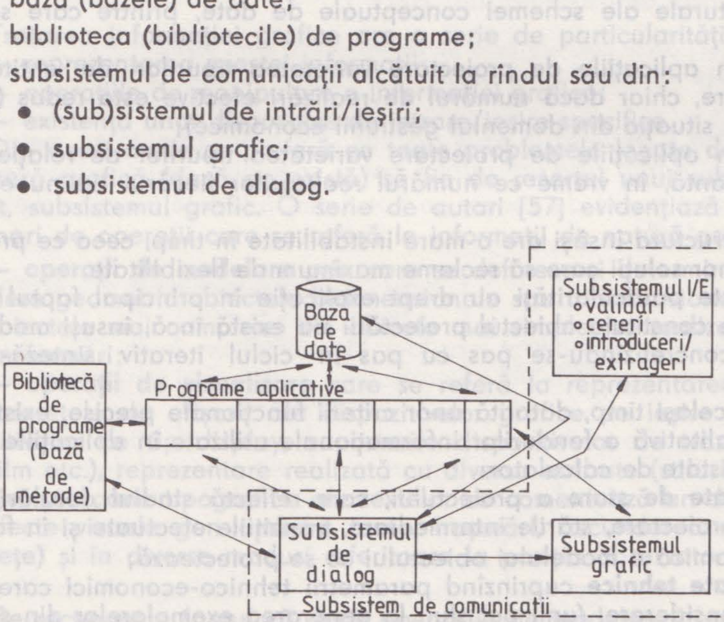


Fig. 18.7. Componente de bază ale unui sistem de proiectare asistată de calculator [57].

Programele aplicative

Inima sistemului de proiectare asistată este constituită de programele aplicative care rezolvă o serie de probleme specifice proiectării, implantînd algoritmi adecvaţi.

La aceste programe se distinge, în general o parte de monitor şi o parte executorie sau funcţională.

Biblioteca (bibliotecile) de programe

Această componentă, care fizic poate fi reprezentată de mai multe entităţi distincte, memorează (în diverse stadii de agregare):

— programe/module aplicative;

— programe/module destinate funcţiilor sistem: gestionarea bazei (bazelor) de date, tratarea subsistemului de intrări/ieşiri, funcţiile specifice subsistemului grafic.

Colecţii de date şi baze de date

Activităţile de proiectare şi în special acelea care sînt automatizate fac apel la o serie de fonduri informaţionale care trebuie să fie gestionate în mod unitar, centralizat, chiar dacă ele în sine şi eventual şi principalele funcţii de control sînt distribuite într-o reţea. În literatura de specialitate se arată că sistemele de gestiune a bazelor de date destinate sistemelor de proiectare asistată de calculator nu au gradul de universalitate al acelora utilizate în aplicaţiile economice, în primul rînd datorită particularităţilor structurale ale schemei conceptuale de date, printre care se amintesc [56]:

a) în aplicaţiile de proiectare varietatea tipurilor de entităţi este foarte mare, chiar dacă numărul de realizări efective este redus (în comparaţie cu situaţia din domeniul gestiunii economice);

b) în aplicaţiile de proiectare varietatea tipurilor de relaţie este şi ea importantă, în vreme ce numărul realizărilor fiecărui tip nu este prohibitiv;

c) structura însăşi are o mare instabilitate în timp, ceea ce presupune găsirea unor soluţii care să reclame maximum de flexibilitate.

Aceste particularităţi au drept explicaţie în principal faptul că produsul care constituie obiectul proiectării nu există încă, însuşi modelul său abstract concretizîndu-se pas cu pas în ciclul iterativ sinteză-analiză-evaluare.

În acelaşi timp, datorită unor criterii funcţionale precise, există o delimitare calitativă a fondurilor informaţionale utilizate în aplicaţiile de proiectare asistate de calculator:

— **date de stare** a proiectului, care reflectă stadiul derulării procesului de proiectare, stările intermediare, tranziţiile efectuate şi în fapt imaginea dinamică a modelului obiectului ce se proiectează;

— **date tehnice** cuprinzînd parametrii tehnico-economici care trebuie luaţi în considerare, (unii pot sluji la generarea exemplarelor din schemele conceptuale);

– **date documentare** care se referă la informaţii bibliografice despre domeniul respectiv;

– **cunoştinţe** dacă se utilizează mijloace ale inteligenţei artificiale ori sisteme expert.

Chiar în ceea ce priveşte fondul de informaţii pentru datele de stare, se remarcă existenţa unor neomogenităţi, datorită (uneori) a necesităţii de a trata concomitent informaţii grafice (referitoare la aspectul vizual al obiectelor) cu informaţii nongrafice (dimensiuni, comentarii, adnotaţii etc.). Pe plan mondial şi în ţară se desfăşoară cercetări intense privind realizarea de baze de date unitare în care aceste aspecte să fie reprezentate şi manipulate concomitent, dar rezolvarea definitivă a problemei constituie încă un obiectiv al viitorului.

Subsistemul de intrări/ieşiri

În fondurile informaţionale componente ale bazei (bazelor) de date se introduc date noi care trebuie **validate** din punct de vedere al completitudinii şi consistenţei. Validarea se poate face în mai multe momente de timp: la introducere, la extragere (deci "post mortem") sau la nivelul întregii baze de date, după un şir de tranzacţii de intrare (pe durata acestei operaţii fiind blocat accesul la baza de date).

În acelaşi timp subsistemul de intrări/ieşiri conţine şi modulele ce îndeplinesc funcţiile de satisfacere a unor cereri complexe de informare (exprimate eventual într-un limbaj specializat).

Subsistemul grafic

Tratarea informaţiei grafice are o serie de particularităţi legate de:
– reprezentarea acestei informaţii;
– operaţiile de manipulare a informaţiei grafice;
– existenţa unor dispozitive de intrare/ieşire specifice.

Din acest motiv se preferă ca toate problemele legate de informaţia de natură grafică (dacă ea există) să fie de resortul unui subsistem specializat, subsistemul grafic. O serie de autori [57] evidenţiază două categorii mari de operaţii care se referă la informaţii de natură grafică:

– **operaţii de modelare** prin care se definesc şi determină din punct de vedere geometric obiectele bidimensionale sau tridimensionale, se compun obiectele mai complexe din obiecte mai simple, se realizează o serie de transformări;

– **operaţii de vizualizare** care se referă la reprezentarea obiectelor pe suprafeţele de afişare ale dispozitivelor grafice de ieşire (ecrane ale dispozitivelor de tip display, suporturile dispozitivelor de trasat – hîrtie, microfilm etc.), reprezentare realizată cu diverse atribute (stiluri de linii, de marcaje, şabloane tipografice ale textelor ce comentează un desen, culori, alte efecte vizuale gen umbriri, tente, haşurări, ascunderea de linii ori suprafeţe) şi în diverse moduri referitoare la poziţia observatorului faţă de obiect.

Se recunoaşte faptul că dacă operaţiile de modelare depind puternic de specificul aplicaţiei, deci de problema concretă de rezolvat, operaţiile de

vizualizare au un caracter de universalitate care a permis standardizarea lor în vederea realizării unor subsisteme nucleu grafic (abordări gen SYGRAPH-CORE, ISO-GKS, X3H3-PHIGS etc.). Aceste variante diferă în principal doar în modul în care se definesc şi identifică *funcţiile grafice elementare*, precum şi aşa numita *bază de date a procesului grafic*, deci informaţiile care permit intercondiţionarea şi înlănţuirea coerentă a funcţiilor într-un context dat.

Subsistemul de dialog serveşte conversării între utilizator şi componentele sistemului de proiectare asistată de calculator. Orice dialog se desfăşoară într-o formă convenţională specifică asupra căreia au căzut de acord cei doi parteneri, formă cunoscută sub numele de **limbaj.** Întocmai ca şi în cazul sistemelor de operare, acest limbaj apare în principal în două forme:

– *limbaj de comenzi* în relaţia utilizator sistem; se recomandă [57] ca un asemenea limbaj să fie cel puţin de puterea unui subset al limbajului de comenzi al sistemului de operare gazdă; unele variante (ex. Ki [57]) permit posibilitatea extinderii libere de către utilizator a acestui limbaj;

– *limbaj de mesaje* în relaţia utilizator sistem.

Foarte răspîndite la ora actuală sînt aşa-numitele limbaje *„prietenoase"* care permit un acces lesnicios al utilizatorului neinformatician. Sînt răspîndite sisteme de dialog de tip *menu ierarhizat* prin care utilizatorul poate să-şi aleagă pe niveluri, funcţii şi în cadrul acestora subfuncţii, cărora ulterior să le furnizeze argumentele adecvate. Se pot prevedea şi sisteme tip *ghid utilizator* adaptabile după gradul de experienţă al utilizatorului. De asemenea, întocmai ca la unele sisteme de operare, se pot prevedea funcţii de asistenţă (tip "HELP").

Interfeţe în cadrul unui SPAC

Interfaţa e definită în sens restrîns [55] drept totalitatea metodelor şi echipamentelor prin care utilizatorul poate comunica cu un sistem de calcul.

Din acest punct de vedere se pot defini *interfeţe funcţionale* de tipul celor din fig. 18.8.

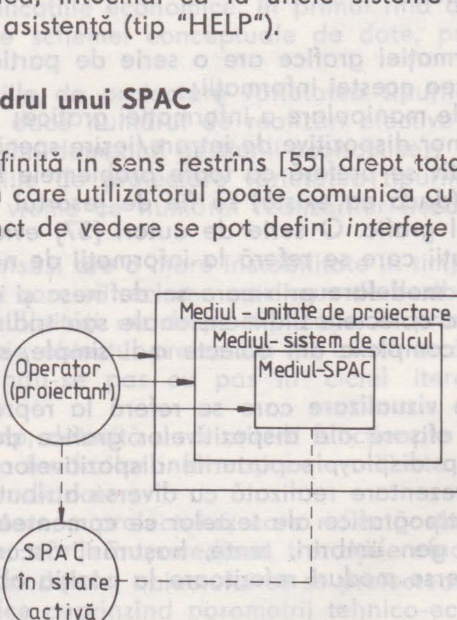


Fig. 18.8. Interfeţe specifice invocării unui SPAC [56].

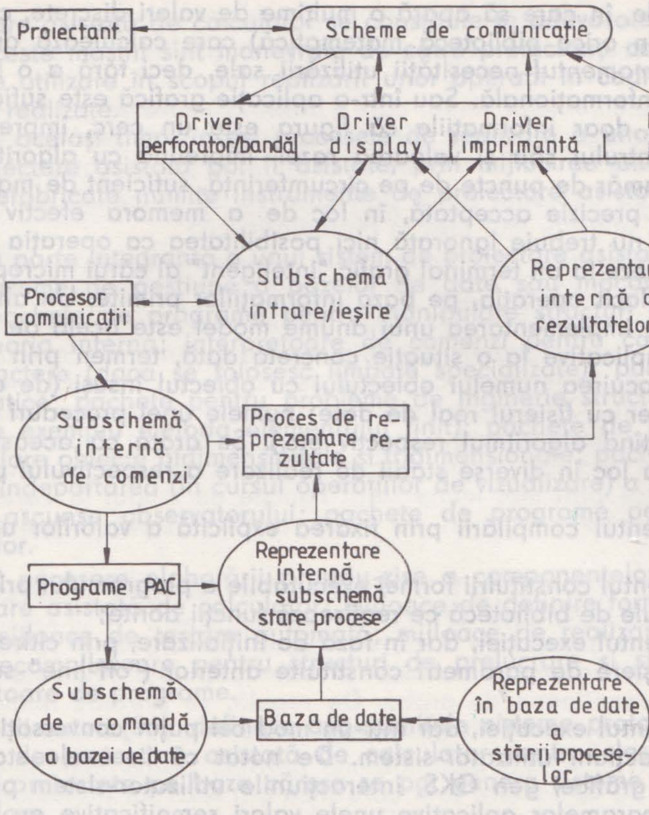


Fig. 18.9. Interfeţe între componentele de bază ale unui SPAC [56].

Într-un sens mai larg interfaţa reprezintă totalitatea mijloacelor de transformare a schemelor de reprezentare a datelor potrivit viziunii particulare a fiecărei componente. Această interpretare poate fi ilustrată de fig. 18.9.

Modele ale obiectului proiectat

Există [57] două modalităţi de realizare a modelelor obiectului proiectat: a) modelarea sub forma unei structuri de date, b) modelare algoritmică.

a) **Modelul tip structură de date** este un model compatibil cu schemele/subschemele bazelor de date, punînd în evidenţă entităţile, relaţiile dintre acestea, modurile de reprezentare în memoria internă şi externă, de asemenea posibilităţile de manipulare a diverşilor reprezentanţi ai claselor de structuri definite.

b) **Modelul algoritmic** se referă în principal la reprezentarea unui algoritm de generare a exemplarelor unei structuri. Astfel de exemplu, pentru o funcţie trigonometrică se va înlocui o reprezentare "punctuală" sub

forma unei tabele, în care să apară o mulţime de valori discrete, cu algoritmii (existenţi în orice bibliotecă matematică) care calculează orice valoare *numai în momentul* necesităţii utilizării sale, deci fără a o memora într-o structură informaţională. Sau într-o aplicaţie grafică este suficient ca să se memoreze doar informaţiile că figura este un cerc, împreună cu coordonatele centrului său şi valoarea razei, împreună cu algoritmul de calcul al unui număr de puncte de pe circumferinţă, suficient de mare pentru a asigura o precizie acceptată, în loc de a memora efectiv aceste puncte. Desigur nu trebuie ignorată nici posibilitatea ca operaţia de trasare să se efectueze la un terminal grafic "inteligent" al cărui microprogram să asigure nemijlocit operaţia, pe baza informaţiilor primite. Un alt aspect legat de fapt de implementarea unui anume model este acela de cuplare a programelor aplicative la o situaţie concretă dată, termen prin care se înţelege [57] înlocuirea numelui obiectului cu obiectul însuşi (de exemplu numele unui fişier cu fişierul real de date; numele unei proceduri cu procedura reprezentînd algoritmul respectiv etc.). Se arată că această înlocuire poate avea loc în diverse stadii de realizare a respectivului program aplicativ:

— în momentul compilării prin fixarea explicită a valorilor unor parametri;

— în momentul constituirii formei executabile a programului prin includerea unor module de bibliotecă ce realizează funcţii dorite;

— în momentul execuţiei, dar în faza de iniţializare, prin citirea prealabilă a unor fişiere de parametri constituite anterior ("off-line" sau "on-line");

— în momentul execuţiei, dar într-un mod cel puţin conversaţional pe baza unei interacţiuni utilizator-sistem. De notat că în unele standarde privind aplicaţii grafice, gen GKS, interacţiunile utilizator-sistem prin care se comunică programelor aplicative unele valori semnificative au fost valorificate după natura acestor valori (poziţie sau poziţii în spaţiu, valoare reală, indice de variantă, identificator figură, şir de caractere), ele fiind comunicate programului prin manipularea unor dispozitive fizice speciale ataşate postului de lucru utilizator.

Maşini şi instrumente de PAC

Majoritatea specialiştilor în probleme de proiectare asistată de calculator, remarcă existenţa mai multor categorii de personal implicat în această problemă:

— utilizatorii unui sistem de proiectare asistată, respectiv proiectanţii obiectului solicitat de comanda socială materializată în specificaţii;

— elaboratorii de sisteme de proiectare asistată care să fie ulterior puse la dispoziţia utilizatorilor;

— realizatorii de aplicaţii de proiectare asistată de calculator, integrabile unui sistem deja existent.

Însuşi un anume sistem de proiectare asistată de calculator poate fi privit ca un produs nou, trebuind să fie la rîndul său proiectat.

Deoarece în procesul de proiectare se transformă specificaţiile iniţiale în documentaţie de proiectare, prin analogie cu ceea ce se întîmplă în procesele de producţie unele proceduri integrate din cadrul unui sistem de

proiectare asistată de calculator se constituie în adevărate maşini software [57]. Aceste maşini sînt manevrate de către proiectanţi conform instrucţiunilor de utilizare în scopul realizării unor operaţii în cadrul proiectelor ce trebuie realizate.

În acelaşi timp, înseşi procesele de realizare a unor anume sisteme de proiectare asistată pot fi asistate, prin mijlocirea unor produse utilitare prefabricate numite instrumente de proiectare asistată [57]. Acestea pot fi:

a) parte integrantă a unui sistem de proiectare asistată:

sisteme de gestiune a bazelor de date sau măcar a seturilor de date; pachete de programe pentru manipulare structuri de date comune în memoria internă; interpretoare de comenzi pentru comenzi de tip şir de caractere (dacă se folosesc limbaje specializate); pachete de rutine matematice; pachete pentru probleme de inginerie structurală (care folosesc de exemplu metoda elementului finit); pachete de programe pentru manipulare obiecte bidimensionale şi tridimensionale; pachete de programe pentru îndepărtarea (în cursul operaţiilor de vizualizare) a liniilor ori suprafeţelor ascunse observatorului; pachete de programe pentru construirea desenelor.

b) necesare elaborării propriu-zise a componentelor unui sistem de proiectare asistată de calculator: mijloace de definire formală a specificaţiilor; mijloace de testare automată; mijloace de realizare a documentaţiei; precompilatoare pentru structuri de prelucrare şi structuri de date; generatoare de programe.

Unii autori [56] definesc aşa-numitele *sisteme prototip nucleu generalizate* de proiectare asistată de calculator pentru clase tipologice de obiecte proiectate pe baza cărora se pot genera sisteme pentru probleme particulare.

Metode inginereşti în PAC

În opinia majorităţii specialiştilor [57] printre metodele inginereşti abordate în cadrul programelor aplicative parte a unui sistem de proiectare asistată de calculator se remarcă:

1. **Metode geometrice.** Se referă la atributele geometrice şi topologice ale obiectelor incluzînd atît aspectele de modelare cît şi cele de vizualizare. În această categorie intră o serie de probleme: reprezentarea punctelor în plan şi spaţiu; modelarea în plan şi spaţiu (modele tip schelet, prin care se reprezintă muchiile reale sau convenţionale; modele prin suprafeţele exterioare; modele prin operaţii booleane între primitive spaţiale etc.); transformări geometrice (translaţie, rotaţie, scalare, simetrie etc.); transformări de vizualizare tip proiecţie (paralelă, perspectivă, axonometrică); determinarea de suprafeţe ori linii ascunse, funcţie de poziţia observatorului faţă de univers populat cu obiecte şi căruia i s-a ataşat un anume referenţial; determinări de desfăşurătoare, înfăşurătoare etc.; rezolvarea unor probleme de simulare a efectelor luminoase pentru obiecte iluminate de la o sursă dată şi ţinînd cont de proprietăţile de material etc.

2. **Metode numerice.** Se referă la rezolvarea unor sisteme de ecuaţii *algebrice* (nu neapărat liniare) sau *diferenţiale* (inclusiv cu derivate parţiale)

care abordează comportamentul static sau dinamic al obiectului proiectat în mediul său probabil de evoluţie. În această ordine de idei foarte răspîndite sînt [57]:

• *metoda elementelor finite*, care e utilizată mai ales pentru tratarea solicitărilor statice sau dinamice ale structurilor de rezistenţă în construcţii, dar şi în probleme de transmisie a căldurii, chiar mecanica fluidelor;

• *metode cu diferenţe finite* prin care se aproximează ecuaţiile cu derivate parţiale, utilizate mai ales în hidrodinamică;

• *metode spectrale* care se referă la transformare Fourier etc.;

• *metode de simulare* care dau posibilitatea de a urmări (ipotetic) evoluţia în timp a modelului obiectului, într-un mediu care de asemenea este modelat. Desigur rezultatele obţinute depind puternic de corectitudinea ipotezelor formulate în legătură cu modelele statice şi dinamice ale obiectului şi mediului său. Oricum simularea nu oferă soluţii, acestea fiind extrapolate de către proiectant prin interpretarea rezultatelor experienţelor simulate. În timpul simulării comportamentul obiectului este considerat ca un proces ce trece prin mai multe stări. Funcţie de modul în care se iau în considerare aceste modificări de stare, modelele pot fi continui şi discrete.

De regulă pentru a înlesni dialogul în interfaţa utilizator (proiectant) – sistem se prevăd limbaje de simulare.

• *metode de prelucrare a datelor experimentale* care se referă la determinarea unor parametri statistici ai unor populaţii rezultate din investigarea unor eşantioane de probă. De regulă, se prevede ipoteza statistică a unei repartiţii normale (care se verifică totuşi, potrivit unor teste speciale).

• *metode de optimizare* prin care încă din faza de proiectare, dacă există mai multe soluţii posibile se aleg acelea care sînt convenabile potrivit unor criterii date. Desigur, nu totdeauna se poate vorbi de un optim în sens strict matematic. Există mai multe clase de probleme de optimizare [57]:

– *liniare* (cele mai cunoscute), pentru care există multe pachete de programe şi sînt stabiliţi algoritmi precişi (simplex de exemplu) accesibili din multe limbaje de programare;

– *neliniare fără restricţii* care utilizează diverse metode de gradient, metode tip Newton, metode de căutare etc.;

– *neliniare cu restricţii* care de obicei modifică într-un anumit mod funcţiile obiectiv pentru a folosi aceleaşi metode în cazul tipului "neliniar fără restricţii".

3. **Grafică de tip comercial.** Sînt metode de reprezentare a rezultatelor sub o formă grafică, ca: histograme; grafice cu bare; diagrame circulare; prin curbe de nivel; vederi pseudoperspectivă.

Medii de programare în SPAC

În ierarhia de procese de tip mediu pentru (sub)procesele de proiectare asistată de calculator, intră şi mediile de programare utilizate în programele aplicative, componente ale sistemului. În acest sens se pot utiliza mai multe limbaje de programare cum ar fi:

a) FORTRAN (diversele sale dialecte), BASIC, ADA – pentru probleme pur tehnico-ştiinţifice;

b) PASCAL, PL/1, C – dacă este vorba şi de manipularea unor structuri informaţionale mai speciale;

c) LISP, PROLOG – pentru probleme de inteligenţă artificială şi în general de programare logică;

d) diverse limbaje orientate spre problemă (procedurale sau neprocedurale).

În cele ce urmează, se vor prezenta cîteva scurte exemple de programe în BASIC pentru unele probleme particulare întîlnite în programele aplicative.

□ Reprezentări geometrice în BASIC

Aplicaţii geometrice

Translaţia în plan. Fie punctul P (x, y) şi un vector de translaţie V de componente tx, ty. Transformatul P′ (x′, y′) va avea coordonatele:

$$x' = x + tx$$

$$y' = y + ty$$

Sub formă matricială, utilizînd coordonatele omogene, rezultă:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

O rutină în BASIC care să efectueze transformarea este:

```
1000 I SUBRUTINA TR2D X, Y, D1, D2, X1, Y1
1010 I X, Y COORDONATELE INIŢIALE
1020 I X1, Y1 COORDONATELE FINALE
1030 ·I D1, D2 DEPLASĂRILE PE AXA ABSCISELOR RESPECTIV ORDONATELOR
1040 DIM U(3), V(3) ! VECTORII DE POZIŢIE OMOGENI
1050 MAT U=ZER(3) ! INIŢIALIZAREA CU ZERO
1060 MAT V=ZER(3) ! VECTORII U ŞI V
1070 LET U(1)=X ! CONSTRUIRE VECTOR IN
1030 LET U(2)=Y ! COORDONATE
1090 LET U(3)=1 ! OMOGENE
1100 MAT F=ZER(3,3) ! INIŢIALIZARE MATRICE DE TRANSFORMARE
1110 LET F (1,1)=F(2,1)=F(3,3)=1 ! INIŢIALIZARE DIAGONALĂ CU 1
1120 LET F(1,3)=D1 ! FACTOR TRANSLAŢIE X
1130 LET F(2,3)=D2 ! FACTOR TRANSLAŢIE Y
1140 MAT V=F * U ! CALCUL VECTOR POZIŢIE
1150 LET X1=V(1)
1160 LET X2=V(2)
1170 RETURN
```

Rotaţia în plan în jurul originii. Fie θ unghiul de rotaţie în jurul originii. Utilizînd coordonatele omogene:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Cos}\,\theta & -\text{Sin}\,\theta & 0 \\ \text{Sin}\,\theta & \text{Cos}\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rutina în BASIC arată astfel:

```
1200 ! SUBRUTINA ROT2D X, Y, G X1, Y1
1210 ! X, Y COORDONATE INIŢIALE
1220 ! X1, Y1 COORDONATE TRANSFORMATE
1230 ! G UNGHI DE ROTAŢIE
1240 DIM S(3), Q(3) ! VECTORII DE POZIŢIE IN COORD. OMOG
1250 MAT S=ZER(3) ! INIŢIALIZARE
1260 MAT Q=ZER(3) ! VECTORI OMOGENI
1270 LET S(1)=X ! FIXARE VALOARE
1280 LET S(2)=Y ! VECTOR
1290 LET S(3)=1 ! INTRARE
1300 LET T=G/57.2957795 ! CONVERTIRE UNGHI IN RADIANI
1310 MAT A=ZER (3, 3) ! INIŢIALIZARE MATRICE TRANSF.
1320 LET A(1,1)=A(2,2)=COS (T)
1330 LET A(2,1)=SIN (T)
1340 LET A(1,2)=-A(2,1)
1350 LET A(3,3)=1
1360 MAT Q=A * S ! CALCUL VECTOR OMOGEN TRANSF.
1370 LET X1=Q(1)
1380 LET Y1=Q(2)
1390 RETURN
```

Arc de cerc trecînd prin trei puncte. Fie trei puncte necoliniare P1, P2, P3. Ele determină un cerc de centru O1. Pe periferie se determină ulterior N puncte echidistante Q1, . . . , QN în eventualitatea aproximării cercului printr-un poligon regulat (dacă dispozitivul care ar trasa arcul nu dispune de interpolator circular). Rutina în BASIC arată astfel:

```
100 ! SUBRUTINA C3P X3(3), Y3(3), N, XO, YO, R, S (), T ()
110 ! X(3), Y(3) COORDONATELE PUNCTELOR INIŢIALE
120 ! N NUMĂRUL DE PUNCTE ECHIDISTANTE DE PE PERIFERIE
130 ! XO, YO, COORDONATELE CENTRULUI (IEŞIRI)
140 ! R RAZA CERCULUI (IEŞIRE)
150 ! S( ) VECTORUL ABSCISELOR DE PE PERIFERIE
160 ! T( ) VECTORUL ORDONATELOR DE PE PERIFERIE
170 MAT A=ZER(3) ! MATRICI DE LUCRU INIŢIALIZATE CU ZERO
180 MAT B=ZER(3)
190 FOR I=1 TO 3 ! COPIERE COORDONATE PUNCTE P1, P2, P3
200    LET A(I)=X3(3)
210    LET B(I)=Y3(3)
220 NEXT I
230 LET D1=A(1) ! FIXARE VECTOR TRANSLAŢIE
240 LET D2=B(1)
250 FOR I=1 TO 3
260        LET X=A(I)
270        LET Y=B(I)
280        LET D1=-X3 (1) ! PREGĂTIRE
290        LET D2=-Y3 (1) ! TRANSLAŢIE IN P1
300        GOSUB 1000 ! APEL RUTINA TRANSLAŢIE 2D IN P1
310        LET A(I)=X1 ! MATRICILE A ŞI B CONŢIN
320        LET B(I)=Y1 ! COPIILE COORDONATELOR P1, P2, P3
330 NEXT I
340 GOSUB 1500 ! DETERMINARE CENTRU CERC TRANSLATAT
```

```
350 LET XO=X3(1)+T1 ! DETERMINARE CENTRU CERC
360 LET YO=Y3(1)+T2 ! ÎN SISTEMUL INIŢIAL
370 LET R=SQR (T1 * T1+T2 * T2) ! RAZA CERC
380 LET D1=T1 ! PREGĂTIRE TRANSLAŢIE IN (XO, YO)
390 LET D2=T2
400 FOR I=1 TO 3 ! TRANSLAŢIE CERC IN
410   GOSUB 1000 ! CENTRUL SĂU
420   A(I)=X1 ! VECTORII A, B CONŢIN
430   B(I)=Y1 ! NOILE ABSCISE ŞI ORDONATE
440 NEXT I
450   DIM W (3) ! VECTOR DE UNGHIURI DE POZIŢIE
460 FOR I=1 TO 3
470   LET L=A(I) ! ABSCISA VECTOR DE POZIŢIE
480   LET H=B(I) ! ORDONATA VECTOR POZIŢIE
490   GOSUB 1700 ! CALCUL UNGHI VECTOR POZIŢIE
500   W(I)=E * 57.29577951 ! GRADE
510 NEXT I
520 IF W(1)<W(2) THEN 580
530 IF W(2)>W(3) THEN 630 ! W(1)>W(2)>W(3)
540 LET W(3)=W(3)-360 ! W(1)>W(2)<W(3) CAZ 2
550 IF ABS (W(3)-W(1))<360 THEN 630 ! TESTARE DACĂ E CAZ 1 CORECT
560 LET W(3)=W(3)+360 ! W(1)>W(2)<W(3) CAZ 2
570 GOTO 630
580 IF W(2)<W(3) THEN 630 ! W(1)<W(2)<W(3)
590 LET W(1)=W(1)-360 ! W(1)<W(2)>W(3) CAZ 1
600 IF ABS (W(3)-W(1))<360 THEN 630 ! TEST CAZ 1 CORECT
610 LET W(1)=W(1)+360
620 LET W(3)=W(3)-360
630 ! SE APELEAZĂ RUTINA TRASARE ARC INTRE UNGHIURILE W(1), W(3)
640 LET C1=W(1)
650 LET C2=W(3)
660 GOSUB 2000
670 LET I%=0 ! PREGĂTIRE
680 LET PO=S(1) ! POZIŢIONARE
690 LET QO=T(1)
700 GOSUB 2500 ! POZIŢIONARE
710 FOR I=1 TO N
720   LET PO=S(I)
730   LET QO=T(I)
740   GOSUB 2500
750 NEXT I
760 RETURN
1000 ! SUBRUTINA TR2D X, Y, D1, D2, X1, Y1
1010 ! X, Y COORDONATELE INIŢIALE
1020 ! X1, Y1 COORDONATELE FINALE
1030 ! D1, D2 COMPONENTE VECTOR TRANSLAŢIE
1040 DIM U(3), V(3) ! VECTORI DE POZIŢIE OMOGENI
1050 MAT U=ZER(3) ! INIŢIALIZARE CU ZERO
1060 MAT V=ZER(3) ! VECTORI U ŞI V
1070 LET U(1)=X ! CONSTRUIRE VECTOR
1080 LET U(2)=Y ! ÎN COORDONATE
1090 LET U(3)=1 ! OMOGENE
1100 MAT F=ZER(3,3) ! INIŢIALIZARE MATRICE DE TRANSFORMARE
1110 LET F(1,1)=F(2,2)=F(3,3)=1 ! INIŢIALIZARE DIAGONALĂ CU 1
1120 LET F(1,3)=D1 ! FACTOR TRANSLAŢIE X
1130 LET F(2,3)=D2 ! FACTOR TRANSLAŢIE Y
1140 MAT V=F * U ! CALCUL VECTOR POZIŢIE TRANSLATAT
1150 LET X1=V(1)
1160 LET X2=V(2)
1170 RETURN
```

```
1500 ! DETERMINARE CENTRU CERC A ( ), B ( ), T1, T2
1510 ! A ( ), B ( ) COORDONATE PUNCTE DE PE PERIFERIE
1520 ! T1, T2 ABSCISA ŞI COORDONATA CENTRULUI FAŢĂ DE REFERENŢIAL
     CURENT
1530 MAT C=ZER (2,2)
1540 MAT D=ZER (2,1)
1550 MAT M=ZER (2,2)
1560 MAT N=ZER (2,1)
1570 FOR i=2 TO 3 ! CONSTRUIRE MATRICE SISTEM ECUAŢII
1580    LET C(i-1,1)=2 * A(i)
1590    LET C(i-1,2)=2 * B(i)
1600    LET D(i-1,1)=A(i) * A(i)+B(i) * B(i) ! ŞI TERMEN LIBER
1610 NEXT i
1620 MAT M=INV(C)        ! SOLUŢIONARE
1630 MAT N=M * D         ! SISTEM
1640 LET T1=N(1,1)       ! MUTARE CENTRU CERC
1650 LET T2=N(2,1)       ! ÎN VARIABILELE T1, T2
1160 RETURN
1700 ! RUTINA ATAN DE CALCUL UNGHI VECTORI POZIŢIE TIP W(i)
1710 ! L, M ABSCISA ŞI ORDONATA VECTORULUI DE POZIŢIE
1720 ! E VALOAREA UNGHIULUI ÎN RADIANI
1730 IF H<0 THEN 1820 ! CADRANELE 3 SAU 4
1740 IF L=0 THEN 1780 ! UNGHIUL ESTE PI/4
1750 IF L<0 THEN 1800 ! CADRANUL 2
1760 LET E=ATN (H/L) ! CADRANUL 1
1770 GOTO 1890
1780 LET E=1.570796327 ! PI/4
1790 GOTO 1890
1800 LET E=3.141592654 - ATN (ABS (H/Y)) ! CADRANUL 2
1810 GOTO 1890
1820 IF L=0 THEN ! UNGHIUL ESTE 3 * PI/4
1830 IF L<0 THEN 1860 ! CADRANUL 3
1840 LET E=6.283185308 - ATN (ABS (H/L)) ! CADRANUL 4
1850 GOTO 1890
1860 LET E=ATN (ABS (H/L))+3.141592654 ! CADRANUL 3
1870 GOTO 1890
1880 LET E=4.712388981 ! 3 * PI/4
1890 RETURN

2000 ! SUBRUTINA ARCA DETERMINARE PARAMETRICĂ COORDONATE ARC
2010 ! XO, YO COORDONATE CENTRU
2020 ! R RAZA CERC
2030 ! C1, C2 UNGHIURI VECTORI POZIŢIE ÎNCEPUT/SFÎRŞIT ARC
2040 LET C3=C1/57.3958 ! TRANSFORM UNGHI ÎN RADIANI
2050 LET P=(C3-C1)/(57.3958 * (N-1)) ! CALCUL INCREMENT ÎN RADIANI
2060 LET CO=COS (P)
2070 LET SO=SIN (P)
2080 LET S(1)=XO+R * COS(C3) ! INIŢIALIZARE VARIABILE
2090 LET T(1)=YO+R * SIN (C3)
2100 FOR I=2 TO N ! CALCUL COORDONATE PUNCTE Q
2110    LET S(I)=XO+(S(I-1)-XO) * CO-(T(I-1)-YO) * SO
2120    LET T(I)=YO+(S(I-1)-XO) * SO+(T(I-1)-YO) * CO
2130 NEXT I
2240 RETURN
2500 ! DAF 2020 RUTINA TRASARE
2510 ! PO, QO COORDONATE PUNCT
2520 ! I%=0 VECTOR STINS; I%=1 VECTOR APRINS
2530 G$=CHR$ (29) ! CARACTER GS PT. TRECERE DAF IN MOD GRAFIC
2540 C9=1023/180.5 ! FACTOR TRANSFORMARE
2550 X%=PO *C9 ! TRANSFORMARE COORDONATE
2560 Y%=QO * C9 ! IN PIXELI
2570 X9%=X%
```

```
2580 Y9%=Y%
2590 K1%=Y%/32 ! DETERMINARE HIY
2600 K2%=Y%-K1% * 32 ! DETERMINARE LOY
2610 K3%=X%/32 ! DETERMINARE HIX
2620 K4%=X%-K3% * 32 ! DETERMINARE LOX
2630 K1%=K1%+32 ! CORECTARE HIY
2640 K2%=K2%+96 ! CORECTARE LOY
2650 K3%=K3%+32 ! CORECTARE HIX
2660 K4%=K4%+64 ! CORECTARE LOX
2670 ! CONSTRUIRE VECTOR DE TRASAT
2680 K$=CHR$ (K1%)+ CHR$ (K2%)+CHR$ (K3%) +CHR$ (K4%)
2690 IF I%=0 THEN 2710 ! VECTOR INTUNECAT
2700 PRINT #1, K$
2710 RETURN
2720 PRINT #1, G$; K$
2730 RETURN
```

SINTEZA 19

Jocuri. Elemente de proiectare și implementare. Cîte ceva despre inteligența artificială. Programe sursă BASIC

□ Puțin . . . despre jocuri pe calculator

Ce joc vă place?

Toată lumea știe că, în general, calculatoarele sînt utilizate pentru lucruri serioase – calcule științifice, economice etc. Calculatorul personal s-a dovedit a fi primul instrument care a permis ca milioane de oameni să-l folosească și pentru jocuri, jocurile pe calculator fiind antrenante, educative și simple. Dacă Bruegel, autorul celebrei picturi "Joc de copii", ar fi fost astăzi printre noi, tabloul său ar fi conținut cu siguranță și calculatoare personale.

După mai mulți autori, lumea jocului se înrudește cu vechea lume gotică a jocurilor, care presupunea un prieten sau o bună tovărășie. Jocurile ne țin companie în modul cel mai serios. Ele pot fi jucate de cei care vor să-și petreacă timpul în mod plăcut; pot participa două sau mai multe persoane, două echipe etc.

Ce fel de jocuri vă place să jucați? Care sînt acele jocuri care ne țin o bună companie? Înainte de a trece la particularitățile realizării jocurilor pe calculatoarele personale, în general, vă invităm să reflectăm puțin asupra tipurilor de jocuri mai des întîlnite.

O primă categorie include fotbalul, hokey-ul, baschetul, baseball-ul și alte sporturi de echipă. Aceste jocuri presupun acțiuni fizice, două echipe și un timp de joc regulamentar (90 de minute, două reprize etc.). Alte jocuri din aceeași categorie sînt: tenisul, golful, înotul etc. Din moment ce cîteva dintre ele sînt jocuri de echipă, toate implică o serie de dispute, fiecare avînd însă un punct de sfîrșit, cum sînt cele din prima categorie.

Altă categorie, pe de o parte jocurile logice – șahul, damele – pe de altă parte jocurile în care hazardul are un rol important (jocul de cărți, monopolul etc.) ș.a.m.d. Și aceste categorii de jocuri au, de asemenea, un sfîrșit dinainte stabilit. Cîteva dintre ele, cum ar fi șahul și damele reprezintă dispute între două persoane. Multe jocuri implică totuși trei sau mai multe persoane care joacă individual.

Mulți dintre dvs. ar fi tentați să se oprească aici, dar noi credem că noțiunea de "joc" este mult mai largă. În exemplele date, sfîrșitul jocului este dinainte stabilit. Ce ziceți însă de jocurile în care participanții joacă

mai degrabă împotriva lor decît între ei? Ce părere aveţi de construirea unui castel de nisip pe plajă care este tot un joc?

O altă categorie de jocuri o reprezintă jocurile simulate pe calculator. Este mult mai ieftin şi mai inofensiv să vă încercaţi combativitatea pe un calculator.

Ce fel de jocuri pot fi jucate pe calculator? Poate că ar fi mult mai firesc să întrebăm care jocuri nu se pot juca pe calculator? Fotbalul, tenisul, boxul, jocul de cărţi pot fi simulate (mai greu sau mai uşor) pe calculator. Fotbalul, de exemplu, este uşor de simulat deoarece, în ciuda alergării înşelătoare a jucătorului dvs. favorit, esenţa jocului constă în mişcarea în sus şi în jos a unui obstacol reprezentat de o linie dreaptă.

Cum să creem un joc?

Jocurile pe calculator nu reprezintă numai programare în BASIC sau în cod maşină. Înainte de a tasta **INPUT**-uri, **POKES**-uri, **PEEKS**-uri, **GOSUB**-uri, trebuie să aveţi o idee foarte clară despre ce va trebui să facă jocul dvs., cum va face şi de ce. Nu încercaţi niciodată să creaţi un joc stînd numai lîngă calculator şi scriind instrucţiuni BASIC sau în cod maşină. Un joc pe calculator reprezintă un program complex şi dacă nu îl gîndiţi "top—down", de la un capăt la celălalt riscaţi să vă pierdeţi ore întregi într-un mod deloc agreabil. Avînd însă un plan bine gîndit înainte de a porni la scrierea primei linii de program, activitatea de programare, de codificare va deveni cu siguranţă cursivă şi plăcută.

Jocurile despre care vă vom vorbi nu reprezintă o chestiune de diagrame şi calcule plictisitoare. Sîntem de principiul că jocurile bune nu sînt construite numai inginereşte, ele sînt, mai ales, create. *Procesul de creare a unui joc reprezintă un fel de vis cu ochii deschişi.* Vizualizaţi figurile, schemele, obiectele ce urmează să apară pe ecran. Gîndiţi-vă la modul în care se mişcă ele, la modul în care sînt comandate. Ce se întîmplă atunci cînd au loc coliziuni? Este ca şi cum vă aplecaţi pe spate în scaunul dvs. moale şi petreceţi cîteva ore spunîndu-vă singuri poveşti. Sau, altfel spus, încercaţi un joc care nu a fost încă programat.

Puteţi crea jocuri din imaginaţia proprie sau inspirîndu-vă din alte surse. Nu este nimic rău dacă vă adaptaţi ideile din alte jocuri dar, în final, jocul trebuie să devină al dvs. Este natural ca, privind la un alt joc să vă întrebaţi: "De ce nu s-a făcut şi asta?". Indiferent de unde luaţi ideea jocului, căutaţi să surprindeţi cît mai multe elemente. Şi, deoarece aţi cunoscut şi iubit (şi probabil urît) mult unele jocuri pe calculator, puteţi împrumuta sau evita aspectele din munca acelor autori de jocuri, dacă consideraţi că trebuie sau nu.

Povestea jocului. Multe din jocurile cu acţiune rapidă au la bază o poveste. Urmează să o extindeţi şi să vă gîndiţi la modul de transpunere a ei într-un joc. Gîndiţi mai multe variante, argumentaţi-vă fiecare decizie şi nu neglijaţi simplitatea. Cel mai dificil lucru este să te exprimi simplu. Evitaţi complicaţiile pe cît posibil.

Mecanica jocului, simulare, coliziuni. Trebuie, de asemenea, avute în vedere: mecanica jocului – ce controlează jucătorul în cadrul jocului şi în

ce mod?; simularea – în ce mod corespunde jocul cu situaţia reală; coliziunile – ce se întîmplă cînd figura jucătorului loveşte în ceva de pe ecran? etc.

Răsplata şi pedeapsa jucătorului. Jocurile sînt ca viaţa; urmăriţi regulile deoarece cînd o faceţi – este bine, cînd nu – este rău. Răsplata cea mai obişnuită este scorul – se obţine de fiecare dată mai mare atunci cînd faceţi lucruri "bune". Există şi alte recompense, ca, de exemplu, acordarea unor distincţii, popularizarea jucătorului etc.

Comunicarea. Jucătorul are nevoie să obţină o serie de informaţii în timpul jocului.

Condiţiile de cîştig/pierdere. Evident, jocul trebuie să se termine odată. Este necesar să hotărîţi ce condiţii sfîrşesc jocul şi atunci verificaţi, prin intermediul unei variabile, din timp în timp să vedeţi dacă aceste condiţii sînt îndeplinite.

Animaţia. Dacă sînteţi o fire ambiţioasă, puteţi adăuga jocului dvs., pentru a accentua mai mult realismul, elemente de animaţie. Jocul s-ar putea juca la fel de bine şi fără aceste secvenţe animate, dar atunci ceva din plăcere ar dispare.

Programarea. Aceasta se întîmplă cînd amuzamentul se transformă în lucru efectiv. Aţi notat repede ideile dvs., aţi testat jocul cu imaginaţia proprie, sînteţi mulţumit că totul este minunat şi nu mai aveţi răbdare să jucaţi. Acum ar trebui să vă opriţi şi să lucraţi altceva, fără amuzament. Începeţi programarea prin a stabili memoria ecran necesară, bucla principală a programului, ce elemente trebuie să cuprindă bucla principală etc. Veţi programa mult mai uşor dacă, înainte de a începe să codificaţi, v-aţi decis asupra subrutinelor de care veţi avea nevoie şi asupra amplasării lor în cadrul programului.

Elemente de proiectare a jocurilor

Definirea lumii jocurilor nu este chiar o chestiune inutilă pentru cei care vor să creeze jocuri originale pe un calculator personal. Este esenţial să înţelegeţi caracteristicile fiecărui joc, înainte de a putea să îl descrieţi pe calculator. Să listăm în cele ce urmează cîteva din întrebările pe care este bine să vi le puneţi înainte de-a vă apuca să concepeţi jocul:

1. Care este numărul jucătorilor implicaţi în joc? Partenerul de joc este calculatorul sau unul din prieteni?
2. Jocul are un punct de sfîrşit sau va fi un joc cu o activitate deschisă (de exemplu, construirea unui castel de nisip, într-o zi de vacanţă, unde plăcerea propriu-zisă constă numai în execuţia lui)?
3. Dacă jocul are un final dinainte stabilit, care va fi condiţia de oprire a jocului (de exemplu, într-un joc de tragere la ţintă, condiţia de sfîrşit este ochirea tuturor ţintelor)?

4. Care sînt factorii (variabilele) determinanţi pentru ca jocul să înainteze? (De exemplu, în fotbal, scorul reprezintă o variabilă, la fel şi timpul de joc rămas, apoi poziţia din teren etc.; într-un joc de tragere la ţintă numărul ţintelor ochite şi numărul ţintelor rămase reprezintă variabile ale programului). Este vital ca toate variabilele (intrare, stare, ieşire) programului să fie de la început identificate.

5. Care va fi modul de joc? Vor face jucătorii schimbări? Ce decizii urmează să ia jucătorii pentru a înainta jocul? Care mişcări (mutări) vor fi legale şi care vor fi ilegale?

6. Cît de multe informaţii va primi fiecare jucător? (În şah, ambii jucători îşi văd reciproc piesele de pe tablă; la jocul de poker, jucătorii îşi ascund, totuşi, cărţile pînă la momentul decisiv . . . etc.).

7. Ce surprize, înfrumuseţări, obstacole, efecte grafice şi sonore vreţi să aveţi în jocul dvs.? Aceşti factori pot afecta rezultatul jocului sau pot contribui la îmbunătăţirea calităţii jocului dvs.?

După cum aţi putut constata, clarificarea a tot ceea ce vrea să realizeze jocul dvs., înainte de a trece la scrierea propriu-zisă a programului, este crucială. Vi se cere, într-adevăr, un efort de gîndire dar vă veţi convinge, dacă mai este cazul, cît de necesar este.

Urmează să realizaţi în continuare "road mapping"-ul jocului dvs. care nu este altceva decît o scurtă descriere a jocului pe care doriţi să-l creaţi. Desigur, puteţi utiliza şi instrumentele de proiectare tradiţionale (schema logică, pseudocodul etc.) dar, considerăm că sînt prea detaliate pentru scopurile noastre. "Road mapping"-ul cuprinde o listă generală cu caracteristicile majore ale programului.

Primul pas în realizarea "road mapping"-ului jocului dvs. este să scrieţi răspunsurile la următoarele şase întrebări:

1. Care este subiectul jocului?
2. Care este definiţia victoriei?
3. Cum va înainta jucătorul spre a-şi atinge obiectivul propus?
4. Ce factori, variabile trebuie să cuprindă jocul?
5. Ce decizii trebuie să ia jucătorii în timpul jocului?
6. Ce decizii trebuie să ia calculatorul în timpul jocului?

Următorul pas este scrierea unui scurt scenariu în care să descrieţi acţiunea jocului.

Remarci

- "Road mapping"-ul nu este un program. El reprezintă o descriere generală a modului cum trebuie jucat jocul şi arată ce decizii şi variabile trebuie luate în considerare, fiind un pas important în realizarea jocurilor dvs.
- Dacă jocul dvs. este unul simplu "road mapping"-ul trebuie să fie scurt.
- "Road mapping"-ul, totuşi, poate avea importanţă egală cu cea a programului, indiferent dacă este un joc scurt sau lung.

Întoarcerea la realitate

În lumea reală nu veţi planifica niciodată totul precum în jocurile pe calculator. Vă puteţi gîndi că jocul dvs. are deja un plan, dar, odată ce mintea dvs. lucrează, nu vă opriţi. Veţi avea multe idei în mijlocul programului, veţi intra într-o încurcătură de programare din care nu ştiţi cum să ieşiţi. Vă veţi opri să improvizaţi şi planul va începe să fie tot mai puţin asemănător cu ceea ce v-aţi propus să realizaţi. Totuşi, nu vă necăjiţi! Va ieşi mai bine ca în vechiul plan. Dar fără un plan, fără o proiectare iniţială, multe din eventualele îmbunătăţiri nu ar fi fost posibile. Ideea de proiectare, de realizare a "road mapping"-ului înainte de codificarea propriu-zisă nu este de a construi un dig care să blocheze creativitatea. Trebuie să săpaţi un canal în care să curgă creativitatea dvs. întocmai unui fluviu. Cînd acesta devine prea puternic, el sare peste mal sau creează braţe moarte de apă. Dar, canalul îl ajută să curgă puternic şi adînc. Cînd aveţi o idee clară încotro mergeţi, aveţi şi o şansă mai bună de a o realiza.

Considerăm că, lecturînd şi cele de mai sus, precum şi alte lucrări de specialitate, veţi putea crea propriile dvs. jocuri. Cel mai important lucru este să realizaţi un joc pe care apoi să-l jucaţi, cu plăcere însă! Dacă inima dvs. nu este în el, jocul dvs. nu va fi atît de amuzant. Să nu încercaţi niciodată să înşelaţi credinţa jucătorilor!

Nu are importanţă cît este de strălucitor jocul dvs.; nici unul nu va fi atît de incitant, de satisfăcător şi de amuzant de jucat decît acela pe care l-aţi creat cu propriul efort.

Cîte ceva despre inteligenţa artificială

În final, ne va face plăcere să menţionăm pe scurt cîte ceva despre o arie fascinantă a programării care, mai tîrziu, o să vă atragă din ce în ce mai mult: *inteligenţa artificială*.

Inteligenţa artificială joacă un rol important pentru programele mai multor tipuri de calculatoare. Calculatoarele care joacă şah constituie un bun exemplu în acest sens. Astfel, unul din cele mai puternice calculatoare construite vreodată – CRAY-XMP, care a cîştigat campionatul mondial de şah programat în 1984, este capabil să examineze 10 milioane de poziţii înainte de a muta şi totuşi... nu este capabil să învingă un mare maestru! Ca să rămînem tot la şah, se estimează că numărul poziţiilor posibile este de ordinul 10^{120}. Dar, un bun jucător îşi reduce problema alegerii mutării următoare la un număr acceptabil şi consideră doar în jur de 100 poziţii care corespund celor cu perspective în evoluţia partidei.

Inteligenţa artificială se referă la simularea de către calculator a procesului gîndirii umane. Termenul a fost pentru prima dată folosit în anul 1956 de prof. John McCarthy de la Universitatea Standfod, California şi acceptat unanim la ora actuală.

Iniţial, obiectivele inteligenţei artificiale erau foarte ambiţioase dar ceea ce s-a reuşit să se pună efectiv pe calculator, după cîţiva zeci de ani de muncă susţinută, a fost partea formalizabilă a inteligenţei omeneşti.

La ora actuală, cercetările în domeniul *inteligenţei artificiale* sînt canalizate pe următoarele teme de bază:

Demonstrarea teoremelor. Demonstraţiile sînt obţinute pe baza axiomelor şi cu ajutorul unor reguli bine definite. Calculatoarele pot constitui în acest caz un instrument ajutător, dar, pînă la ora actuală, nivelul de dezvoltare nu permite demonstrarea automată a teoremelor care nu pot fi demonstrate de om. Un succes remarcabil a fost demonstrarea faimoasei teoreme care spune că orice hartă politică poate fi colorată utilizînd doar patru culori, astfel încît două ţări învecinate să nu fie colorate la fel. În anul 1976, K. Appel şi W. Haken, rulînd 1 200 ore pe calculatorul Universităţii din Illinois, au reuşit să reducă teorema la un număr foarte mare (dar finit) de cazuri particulare mai simple, efectuînd apoi demonstraţia pentru fiecare caz în parte. Bizareria situaţiei, din punctul de vedere al unui matematician, constă în imposibilitatea de a reface la masa de lucru, în vederea verificării, şirul enorm de raţionamente parcurs de maşină.

Jocuri. Cercetările sînt direcţionate către găsirea acţiunii optime într-o situaţie de joc şi elaborarea strategiei cîştigătorului.

Roboţi. Cercetările din acest domeniu sînt direcţionate în principal către elaborarea metodelor de conducere a manipulatoarelor care efectuează funcţiuni complexe, obţinerea unor traductoare, receptoare şi a unor limbaje de nivel înalt necesare descrierii condiţiilor de lucru şi a comenzilor de conducere. Exemple: ochii, urechea artificială, mîini artificiale etc.

Recunoaşterea configuraţiilor. Problema constă în instruirea calculatorului de a recunoaşte fizionomiile, mediul înconjurător, fotografiile etc. S-au elaborat deja o serie de metode de analiză a imaginilor, astfel încît calculatorul să poată recunoaşte imagini destul de complexe, în condiţiile în care, iniţial, se introduc informaţiile necesare. Dar, nu există încă o metodologie generală pentru recunoaşterea unui spectru larg de obiecte.

Înţelegerea limbajului natural şi a vorbirii. Scopul cercetărilor în acest domeniu îl constituie învăţarea calculatorului să înţeleagă limbajul uman natural, ca de exemplu limba japoneză, engleză sau franceză. Există deja anumite sisteme experimentale care înţeleg limba engleză şi japoneză, este drept, numai în cazul în care numărul subiectelor este limitat. Dacă numărul lor creşte, numărul cuvintelor şi frazelor necesare creşte şi mai repede, ceea ce necesită introducerea „bunului simţ" şi a „deducţiei logice". Astfel, aici sînt necesare cercetări suplimentare. La ora actuală, se realizează, de pildă, traduceri cu o precizie de 85–90% în cazul textelor ştiinţifice sau tehnice, la o viteză de 3 000 cuvinte pe oră, utilizînd dicţionare conţinînd pînă la 40 000 cuvinte.

Proiectarea bazelor de cunoştinţe. Cercetările în acest domeniu sînt îndreptate către acumularea de cunoştinţe expert şi găsirea automată a soluţiei problemei puse. Pentru aceasta, se folosesc *sistemele expert*, componentă a inteligenţei artificiale.

Un **sistem expert** este format, în principiu, dintr-un grup de programe şi o colecţie de informaţii de un tip special, cu ajutorul cărora se poate purta un dialog om-calculator, în vederea rezolvării problemelor (de regulă, de natură nematematică) dintr-un domeniu bine circumscris. Aceste sisteme determină reorientarea calculatorului de la prelucrarea obişnuită a datelor (prin operaţii aritmetice) spre prelucrarea inteligentă a cunoştin-

ţelor, prin raţionamente. Dacă materia primă a sistemelor „clasice" o constituie datele, transformate în informaţii, sistemele expert pleacă de la informaţii şi cunoştinţe, creînd din ele alte cunoştinţe.

De realizarea sistemelor expert se ocupă specialişti din două domenii:

a) *ingineria cunoaşterii* (termen japonez evitat, în general, de abordările de sorginte anglo-saxonă) – care elaborează *algoritmi de rezolvare* (succesiune de acţiuni pentru rezolvarea problemei concrete) şi *structura bazei de cunoştinţe.*

b) *specialişti în problemă* – care deţin informaţii factice şi care cunosc regulile deducţiei (informaţia şi regulile deducţiei constituie ceea ce se numeşte "baza de cunoştinţe" a sistemului expert).

Un sistem expert conţine totodată şi module destinate *dialogării prietenoase cu utilizatorul,* inclusiv prin procedee speciale cum ar fi: grafice, limbaj sonor etc., numite la un loc **"interfaţa inteligentă om-maşină".**

La ora actuală, sistemele expert se utilizează în medicină, exploatarea şi repararea aparaturii, în educaţie, explorări geologice, petrol etc.

Cele mai multe probleme, şi în special cele mai interesante, se prezintă sub forma căutării drumului optim care duce de la o stare iniţială la starea finală dorită.

În decurs de 30 de ani, un grup relativ nenumeros de cercetători încearcă cîteodată, cu succes, alteori fără, să elaboreze *programe* care să permită calculatorului să rezolve *raţional* problemele.

La mijlocul anilor '70, după două decenii de progres lent şi abia perceptibil în acest nou domeniu al inteligenţei artificiale, cercetătorii au ajuns la următoarea concluzie fundamentală despre *comportamentul raţional* în general: "acesta necesită o cantitate colosală de cunoştinţe pe care oamenii le posedă ca pe ceva de la sine înţeles, dar care, în timp, trebuie transmise maşinii". *Rolul central* al cunoştinţelor în comportamentul bazat pe raţiune explică de ce pînă în momentul de faţă cel mai mare succes l-au avut programele care constituiau nişte *„experţi"* în domenii specializate foarte înguste. Vis-a-vis de aceasta, la început, cînd s-a încercat elaborarea *„sistemelor universale de rezolvare a problemelor"* s-a presupus că principala componentă a intelectului o constituie posibilitatea de *deducţie logică.* Dar aceste încercări nu au dat rezultatele aşteptate şi în cele mai multe cazuri, la ora actuală, formalismul logic este combinat cu alte surse ale intelectului.

Din cauza neajunsurilor calculatoarelor actuale care nu permit implementarea rezultatelor obţinute în domeniul inteligenţei artificiale, în momentul de faţă se elaborează aşa-numitele **sisteme din generaţia a V-a.** Se pune un accent deosebit în acest caz pe prelucrarea cunoştinţelor pe baza unor teorii şi tehnologii noi. Calculatoarele din generaţia a V-a reprezintă un sistem de prelucrare a cunoştinţelor în care nu vor exista limitările tehnologice ale calculatoarelor obişnuite, ceea ce va permite maşinii să interacţioneze inteligent cu omul şi să construiască deducţii pe baza structurii de cunoştinţe încărcate în ea, structură corespunzătoare anilor '90.

Calculatoarele din generaţia a V-a trebuie să satisfacă următoarele funcţiuni de bază:

– *rezolvarea problemelor şi deducţii logice:* deducţia logică şi inducţia aplicate la rezolvarea unei probleme concrete, precum şi formularea de ipoteze pe baza unor informaţii incomplete;

– *gestiunea bazelor de cunoştinţe:* stocarea, păstrarea şi utilizarea diferitelor cunoştinţe necesare pentru realizarea deducţiilor logice;

– *implementarea unei interfeţe inteligente:* realizarea unei interfeţe exterioare cu utilizarea limbajelor naturale (frază, voce), grafică, imagine şi posibilitatea de conversaţie;

– *programare inteligentă:* codificarea automată a algoritmului de rezolvare a problemei date sub forma unor programe executabile.

Pentru atingerea ţelurilor propuse, se studiază şi se elaborează diferite metode de abordare a arhitecturii hardware, a pregătirii necesarului matematic şi a realizării inteligenţei artificiale.

Primul pas către elaborarea mijloacelor de prelucrare a cunoştinţelor şi a calculatoarelor din generaţia a V-a îl constituie alegerea unui *limbaj de programare* cît mai potrivit pentru descrierea activităţii intelectuale. De regulă, un sistem expert se construieşte utilizînd un limbaj specializat de tipul LISP, PROLOG, SNARK etc. Dar, înainte de a alege limbajul, este necesar să fie înţelese mecanismele fundamentale ale activităţii cerebrale. Trebuie subliniat că majoritatea calculatoarelor de astăzi sînt concepute pentru modelarea mecanismelor de bază ale calculelor numerice, adică a celor 4 operaţii aritmetice.

Pentru rezolvarea unor probleme complexe, oamenii folosesc diferite metode care permit scurtarea procesului de căutare a soluţiei pe baza cunoaşterii unor anumite corelaţii. Ei pot utiliza teoreme matematice sau alte reguli mai puţin formale, bazate pe experienţa acumulată, pot partiţiona problema complexă în subprobleme mai simple sau pot judeca prin analogie cu acele probleme care au mai fost rezolvate.

Multe programe elaborate pe parcursul primelor două decenii de cercetări în domeniul inteligenţei artificiale, în marea lor majoritate erau fundamentate pe metodele raţionamentului formal logic. Logica este considerată, de obicei, metoda universală de gîndire. Din toată varietatea formelor ei, *logica predicatelor* este cea mai aproape de limbajul nostru obişnuit, de aceea o poate utiliza oricine, cu toate că, la început, aceasta poate crea dificultăţi pentru cei care nu sînt obişnuiţi cu rigurozitatea gîndirii. Natural, că numai logica predicatelor nu poate să cuprindă toate procesele de gîndire dar, fără discuţie, este cel mai puternic instrument.

Una din metodele cele mai populare ale deducţiei logice este cea bazată pe *tehnica rezoluţiei* care constă din infirmarea negaţiei (demonstraţia "pornind de la contrariu").

Pentru a aplica metoda rezoluţiei, înainte de toate, este necesar ca afirmaţia de demonstrat să fie transpusă în cadrul formalismului logic, prin intermediul calculului cu predicate. După aceea, afirmaţia este negată şi problema se rezolvă împreună cu un set de axiome – afirmaţii absolut adevărate în domeniul respectiv studiat. Dacă combinaţia dintre axiome şi negaţia făcută ne conduce la o absurditate, rezultă că negaţia este falsă şi deci, afirmaţia iniţială – adevărată.

În anul 1964, A. Robinson a demonstrat că metoda rezoluţiei are prioritatea de consistenţă: dacă afirmaţia iniţială este adevărată, atunci în orice caz, mai devreme sau mai tîrziu, această metodă va conduce la o

absurditate. Dacă afirmaţia iniţială este falsă, atunci nu avem garanţia că procesul de rezolvare prin metoda rezoluţiei nu va fi infinit. Rezultatele lui Robinson au pus bazele începutului unor cercetări active, legate de utilizarea metodei rezoluţiei şi a altor metode înrudite în domeniul demonstraţiei automatizate a teoremelor. Dar metoda rezoluţiei are un mare neajuns: se supune fenomenului de "explozie combinatorică" atunci cînd numărul de rezoluţii efectuate de program creşte exponenţial în funcţie de complexitatea problemei.

Deducţia logică este procedura de obţinere a unei informaţii necunoscute pe baza cunoştinţelor şi datelor avute. Pentru a înţelege esenţa oricărei probleme, trebuie efectuată o deducţie logică, cu toate că nu întotdeauna sîntem conştienţi de acest fapt. La baza proceselor de gîndire stau un volum oarecare de cunoştinţe şi posibilitatea de a face deducţii logice.

Deducţiile logice se fac după anumite reguli, principala regulă fiind regula construcţiei silogismelor. Silogismul este o deducţie de tipul următor: dacă A este B şi B este C, rezultă A este C. Silogismul aritmetic constă dintr-o premisă majoră şi una minoră, ca de exemplu *premisa majoră:* şaizeci de oameni pot face o treabă de şaizeci de ori mai repede decît un singur om; *premisa minoră:* un om poate săpa o gaură de stîlp în şaizeci de secunde. Oare calculatorul poate descoperi greşeala dacă nu este... tras de mînecă?

Să înveţi calculatorul arta de a face deducţii este posibil prin implementarea regulilor deducţiei nemijlocit în aparatură; acest mod de abordare este utilizat acum la realizarea calculatoarelor din generaţia a V-a.

Vulnerabilitatea metodelor logice constă în ceea ce constă şi forţa lor: foarte multe tipuri de cunoştinţe, incluzînd aici şi pe cele false şi pe cele incomplete care sînt atît de proprii majorităţii problemelor din lumea reală, nu pot fi formulate în limitele stricte ale logicii formale.

În ziua de astăzi există zeci de programe complexe care rezolvă probleme complicate din diverse domenii, precum diagnosticarea medicală, planificarea experienţelor genetice, explorarea geologică şi construirea automatizată. Principala sursă în aceste sisteme expert o constituie gîndirea neformală bazate pe cunoştinţe vaste care au fost adunate de la oameni experţi. În majoritatea acestor programe cunoştinţele sînt codificate sub forma a sute de reguli de tipul "dacă-atunci", rezultate din practică. Un astfel de tip de reguli poartă denumirea de *euristice.* Regulile limitează căutarea, atrăgînd atenţia programului către cele mai probabile căi de rezolvare. Mai mult decît atît — şi în aceasta constă diferenţa dintre programele bazate pe euristice şi cele bazate pe metode formale — aceste sisteme expert pot explica modul lor de gîndire într-o formă inteligibilă pentru om. Aceste explicaţii devin posibile mulţumită faptului că rezolvările adoptate de program sînt bazate pe reguli preluate de la oameni-experţi şi nu pe reguli ale logicii formale.

De exemplu, sistemul MYCIN pune diagnosticul în septicemii şi meningită, indicînd şi tratamentul adecvat. Problema rezolvată de acest sistem constă în a determina care din microorganismele posibile a dat naştere îmbolnăvirii şi de a recomanda un tratament pe baza diagnosticului stabilit. Sistemul MYCIN are o bază de cunoştinţe formată din 500 de reguli euris-

tice, în plus fiecărei reguli îi este asociată o mărime care caracterizează gradul de încredere în veridicitatea acestei reguli.

În afară de regulile euristice, sistemele expert își extrag "forța" și din alte surse, ca de exemplu considerentele dictate în exclusivitate de bunul simț la care oamenii rareori se gîndesc. În acest sens, sistemul MYCIN dă din nou un bun exemplu. Posedînd informații inițiale generale despre pacient, sistemul MYCIN în deducțiile sale pleacă de la țelul propus – identificarea organismului generator de boală. El pune întrebări care pot releva simptome specifice care pot confirma ipoteza unui diagnostic. O astfel de orientare spre țelul propus nu înseamnă că algoritmul de luare a deciziilor este implementat în program așa cum consideră cei care nu acceptă situația că sistemele expert prezintă anumite calități intelectuale. Programele de tipul MYCIN, în realitate, se adaptează la situații care nu pot fi prevăzute de programator, acesta neputînd ști în ce mod concret își va utiliza sistemul baza de cunoștințe.

O altă strategie eficientă utilizată de ființele raționale – oamenii și bineînțeles software-iștii este aceea de a împărți o problemă mai complexă în subprobleme mai simple, numită și *„partiționează și decide"*.

Fiecare regulă utilizată de sistemul expert poate fi foarte simplă în sine. Cîteodată, aceste reguli sînt slab organizate sau le lipsește în totalitate orice fel de organizare. Dar, acest sistem de reguli ca un tot unitar este în măsură să rezolve probleme tehnice complicate, demonstrînd prin aceasta competența sa de expert. În acest caz se observă o variantă a fenomenului denumit *sinergie*, cînd intregul devine mai „mare" decît simpla însumare a componentelor sale. Acest fenomen este atît de generalizat încît este abordat ca fiind ceva de la sine înțeles, dar el jucînd unul din cele mai importante roluri în sistemele expert.

Unul din cele mai fructuoase programe avînd caracteristicile inteligenței artificiale a fost și primul sistem expert din punct de vedere istoric. Acest program numit DENDRAL, precum și succesorul lui GENOA sînt larg utilizate în laboratoarele de chimie organică din multe țări. Sistemul DENDRAL stabilește structura moleculelor organice, bazîndu-se pe datele spectrometriei de masă, ale rezonanței magnetice nucleare și a altor tipuri de informații. La fel ca și MYCIN, sistemul DENDRAL este, în esența sa, un sistem de diagnosticare.

În procesul de căutare care constă din sinteză, analiză și valorizare a noilor concepții, sistemele expert sînt conduse de euristice de un caracter destul de general. Amintim în acest sens pe cele de tipul: "Studiul situațiilor extremale", "factorul noroc" etc. euristice care ajută programul-cercetător să își însușească noi concepte.

Astfel, de la utilizarea euristicelor pentru descoperirea unor noi concepte și fapte teoretice și pînă la generarea de euristice noi, pe baza celor vechi, calea nu este lungă. Acest pas este legat de ceea ce este considerat a fi țelul principal în domeniul inteligenței artificiale – elaborarea de programe capabile să se autoinstruiască pe baza experienței pe care o acumulează în timpul lucrului. În ultimii ani, mai mulți cercetători au reușit să elaboreze programe care generează legi generale pe baza experienței acumulate.

Perspectivele realizării unor programe autodidacte inteligente, în mare măsură depind de faptul dacă se vor găsi căi de implementare a unei surse

atît de importante ca aceea care este *analogia.* Utilizarea analogiei este eficientă dacă considerentele comparate într-adevăr posedă elemente comune adică dacă anumite atribute sînt apropiate între ele. Ca de exemplu, la marea majoritate a bolilor avem atribute generale – "cauze", "simptome", "tratament" ş.a. şi de aceea între ele putem face analogii. Aceste analogii sînt folositoare deoarece îmbolnăvirile, care au cauze asemănătoare, necesită deseori tratamente asemănătoare. Chiar aşa stau lucrurile, deoarece studenţii medicinişti învaţă cazuistica prin analogie cu cele învăţare, aşa că şi programele de diagnosticare medicală ar putea face acelaşi lucru.

Totuşi, maşinile nu vor putea să atingă nivelul inteligenţei umane în rezolvarea celor mai interesante probleme pînă cînd programele nu vor deveni asemănătoare cu oamenii, în două privinţe foarte importante:

– posibilitatea de acumulare a propriei experienţe în decursul unui timp îndelungat pe care să o numim *„experienţă de viaţă";*

– posibilitatea de comunicare unul cu celălalt, prin aceasta asigurîndu-se instruirea.

ELIZA – un joc cu inteligenţă artificială

În jocurile cu inteligenţă artificială, jucătorul pune întrebări calculatorului şi aşteaptă un răspuns corespunzător. În primul joc de acest gen, ELIZA, elaborat în anul 1967 de J. Weizenbaum, calculatorul simula un medic care efectua psihanaliza "pacientului" aflat la terminal. Jucătorul tastează un enunţ sau o întrebare, iar calculatorul îi răspunde. Iată pe scurt o posibilă conversaţie:

(ELIZA) – Cum vă simţiţi astăzi?
(JUCĂTORUL) – Nu prea bine!
(ELIZA) – De ce nu vă simţiţi bine?
(JUCĂTORUL) – Pentru că sînt îngrijorat.
(ELIZA) – De ce sînteţi îngrijorat?
(JUCĂTORUL) – Pentru că cineva este în spatele meu.
(ELIZA) – De ce se află în spatele dvs.?
(JUCĂTORUL) – Pentru că vor să mă atace!
(ELIZA) – De ce vor să vă atace?
(JUCĂTORUL) – Pentru că nu le plac.
(ELIZA) – De ce nu le plăceţi?
(JUCĂTORUL) – Pentru că sînt din alt spaţiu.
(ELIZA) – De cît timp aveţi aceste gînduri?

J. Weizenbaum, autorul programului, a făcut un experiment cu jucători care nu ştiau dacă partenerii lor sînt calculatoare sau oameni.

Fiind solicitaţi să ghicească cu cine au conversat, a reieşit clar faptul că jucătorii nu au fost în stare să spună dacă s-au întreţinut cu o persoană sau cu o maşină.

Construcţia acestor jocuri este, evident, dincolo de scopul acestei cărţi. Programele de inteligenţă artificială ar fi o temă bună pentru fiecare dintre dvs. Dacă acest subiect vă place, vă rugăm să nu vă terorizaţi calculatorul. În realitate, el poate fi o maşină prietenoasă.

Menţionăm că, folosind un calculator personal şi utilizînd limbajul BASIC (de ex. BASIC Microsoft) puteţi descoperi, progresiv, cîteva din noţiunile fundamentale care stau la baza conceptului de inteligenţă artificială: recursivitate, reprezentarea cunoştinţelor, strategii de căutare etc. Vă recomandăm să faceţi aceasta în cadrul unor programe de jocuri cu inteligenţă artificială [66, 67].

☐ Jocuri în BASIC.25 programe sursă

Jocuri pe HC-85, TIM S, SPECTRUM

FORMULA 1. Vă invităm la cel mai pasionant raliu Grand Prix. Mașinile sînt aliniate la linia de start. Titlul de campion al lumii pentru Formula 1 este pus în joc. Toți piloții nu vizează decît primul loc. Se fac ultimele verificări. Drapelul de start a fost coborît, mașinile au pornit în cursă. Controlați mașina dvs. cu ajutorul tastelor de deplasare a cursorului. În timpul cursei . . .

```
  10 REM FORMULA 1
 100 REM PROGRAM PRINCIPAL
 110 GO SUB 1000
 120 PRINT AT 0,5 "sc="; sc; "TIMP RESTANT="; ti : GO SUB 300
 135 GO SUB 500
 140 PRINT AT rc, cc; " "
 150 PRINT AT 21,0 : POKE 23692, - 1 : PRINT
 160 GO SUB 600
 170 LET ti=ti-1
 180 IF ti>=0 THEN GO TO 120
 190 PRINT AT 0,5; "REZULTATUL"; sc
 200 PRINT AT 1,5; "————————————"
 210 INPUT "JUCAȚI ? (d/n)" ; y$
 220 IF y$="d" THEN GO TO 110
 230 STOP
 300 REM PISTA
 310 IF (xw+dw)<8 OR (xw+dw+ww)> 240 THEN LET dw=0 : LET lw=0
 320 LET lw=lw-1
 330 PLOT xw, yw : DRAW dw, -8
 340 PLOT xw+ww, yw : DRAW dw, -8
 350 LET xw=xw+dw
 360 IF lw>0 THEN RETURN
 370 LET lw=INT (RND * 4)+2
 380 LET dw=INT (RND * 17) -9
 390 RETURN
 500 REM OBSTACOLE
 510 LET ga=RND * 150+sc
 520 IF ga<150 THEN RETURN
 530 LET co=INT (RND * 3+xw/8)+1
 540 IF ga>225 THEN PRINT AT 20, co; INK 3; "3" : RETURN
 550 PRINT AT20, co; INK 5; "0"
 560 RETURN
 600 REM MAȘINA
 610 LET t$=INKEY$
 620 LET cc=cc+(t$="8" AND cc<30)-(t$="5" AND cc>1)
 630 LET c$=SCREEN$ (rc, cc)
 640 PRINT AT rc, cc; "3"
 650 IF c$=" " THEN LET sc=sc+1 : RETURN
 660 IF c$="0" THEN LET sc=sc+10 : BEEP, 1,6 : RETURN
 680 LET sc=sc-25: BEEP 1,1
 690 RETURN
1000 REM INIȚIALIZARE
1010 LET rc=10 : LET cc=15
1020 LET xw=110 : LET yw=16
1030 LET lw=8 : LET dw=0
1040 LET sc=0 : LET ww=40
```

```
1050 LET ti=200
1060 RESTORE : RANDOMIZE : CLS
1070 FOR k=0 TO 7
1080   READ c
1090   POKE USR "a"+k, c
1100 NEXT k
1110 FOR k=0 TO 7
1120   READ c
1130   POKE USR "b"+k, c
1140 NEXT k
1150 PLOT xw, rc * 8 : DRAW 0, -rc * 8
1160 PLOT xw+ww, rc * 8: DRAW 0, -rc * 8
1170 PRINT AT 0,5; FLASH 1; "ATENŢIE!"
1180 FOR i=1 TO 6: BEEP . 5, i: NEXT i
1190 RETURN
1200 DATA 24, 68, 68, 24, 24,24, 68, 68
1210 DATA 68, 68, 24, 24, 24, 68, 68, 24
```

MINI-MANCALA. Jocul creat de C. Freelink are la bază un joc arab foarte vechi. El constă în trecerea pietrelor dintr-o cupă într-alta. Există în total patru cupe (A, B, C, D), dintre care două (A şi B) aparţin calculatorului. La începutul jocului fiecare cupă conţine cîte două pietre. Un jucător poate lua pietrele din propria cupă şi să le distribuie în cele trei cupe în sens invers acelor de ceasornic. Aţi cîştigat atunci cînd toate pietrele se află în cupa dvs.

Pe calculator cupele sînt vizualizate prin căsuţe pe care sînt înscrise cifre reprezentînd numărul de pietre pe care le conţine fiecare cupă.

Puteţi opta pentru unul din cele trei nivele de dificultate (nivelul 1 este cel mai simplu). Puteţi, de asemenea, decide dacă doriţi sau nu să fiţi primul jucător. Calculatorul vă va întreba în care cupă preferaţi să vă plasaţi pietrele. Să nu fiţi surprinşi dacă acest joc complicat a fost realizat printr-un program atît de scurt.

```
 10 REM * MINI MANCALA *
 20 DIM m (2,2) : DIM l (4,2)
 30 GO TO 700
 99 REM * INIŢIALIZARE *
100 FOR i=1 TO 2
110   LET m(1, i)=2 : LET m(2, i)=2
120   FOR j=1 TO 4
130   READ x : LET l (j, i)=x
140   NEXT j
150 NEXT i
160 LET turn=0: LET xturn=0 : RETURN
170 DATA 1, 2, 2, 1, 1, 1, 2, 2
199 REM * AFIŞARE TABLOU *
200 CLS : PRINT AT 8, 14; "2 2"
210 PRINT AT 10, 14; "2 2"
212 PRINT AT 4, 10; "MINI MANCALA";
214 PRINT AT 7, 12; "A B";
216 PRINT AT 11, 12; "C D";
220 FOR i=0 TO 4 STEP 2
230 PLOT 108+i * 8, 115
240 DRAW 0, -32
250 PLOT 108, 115-i * 8
260 DRAW 32, 0: NEXT i
265 RETURN
299 REM * INTRARE *
300 PRINT AT 20, 7; "DUMNEAVOASTRĂ"; TAB 31; " ";
```

```
310 INPUT "LUAŢI PIETRE DIN";
    TAB 32; "C SAU D?"; x$;
330 IF (x$="C" OR x$="c"
    AND m (2, 1)>0 THEN LET l=2: RETURN
340 IF (x$="D" OR x$="d")
    AND m (2, 2)>0 THEN LET l=3: RETURN
350 BEEP 1,3
360 GO TO 310
399 REM
400 PRINT AT 20, 7; "EU"
410 FOR w=0 TO 300: NEXT w
420 LET q=m(1,1) * 1000+m(1,2) * 100+m(2,1) * 10+m(2,2)
430 LET l=4
440 IF m(1,1)>0 AND (lev<3 AND RND * lev<·4) OR m(1,2)=0 OR q=
    1430 OR q=6110 OR q=1160 THEN LET l=1
450 PRINT AT 21, 3; "IAU PIETRELE
    DIN"; CHR$(65+(l=4))
460 RETURN
499 REM
500 LET g=l (l, 1): LET h=l (l, 2)
510 IF m(g, h)=0 THEN RETURN
520 LET l=l+1 : IF l>4 THEN LET l=l-4
530 LET i=l (l, 1) : LET j=l (l, 2)
540 GO SUB 600: GO TO 510
599 REM
600 FOR w=1 TO 200: NEXT w
610 LET m(g, h)=m(g, h)-1: LET m(i, j)=m(i, j)+1
620 PRINT AT 6+2 * g, 12+2 * h; m(g, h);
630 PRINT AT 6+2 * i, 12+2 * j; m(i, j);
640 RETURN
699 REM * PROGRAM PRINCIPAL *
700 GO SUB 100 : GO SUB 200
710 INPUT "NIVEL 1-3: " ; a$
720 IF a$ < > "1" AND a$ < > "2"
    AND a$ < > "3" THEN GO TO 710
730 LET lev=VAL a$: PRINT AT 14, 12; "NIVEL="; lev
740 INPUT "CINE ÎNCEPE? DVS (v) SAU eu (m)? " ; TAB 32; a$
750 IF a$="V" OR a$="v" THEN GO TO 800
760 IF a$ < > "m" AND a$ < > "M" THEN GO TO 740
770 FOR w=0 TO 200: NEXT w: GO SUB 300: GO SUB 500
780 LET turn=turn+1; IF m(1,1)+m(1,2)=0 THEN PRINT AT 20, 6;
    "CIŞTIGAT ÎN"; turn+xturn; "LOVITURI"; TAB 31;
    " "; TAB 31; " ": GO TO 830
790 IF turn=40 THEN GO SUB 900: IF turn=40 THEN GO TO 830
800 FOR w=0 TO 200: NEXT w : GO SUB 400: GO SUB 500
810 LET turn=turn+1: IF m(2,1)+m(2,2)=0 THEN PRINT AT 20,6;
    "CIŞTIGAT ÎN"; turn+xturn; "ÎNCERCĂRI"; TAB 31;
    " "; TAB 31; " "; : GO TO 830
820 IF turn=40 THEN GO SUB 900: IF turn=40 THEN GO TO 830
825 GO TO 770
830 INPUT "CONTINUAŢI?"; TAB 32; " (d/n): " ; a$
840 IF a$="D" OR a$="d" THEN RUN
850 IF a$="n" OR a$="N" THEN STOP
860 GO TO 830
900 INPUT "AM SĂRIT UN TUR.
    CONTINUAŢI (d/n) : " ; a$
910 IF a$="D" OR a$="d" THEN LET xturn=
    xturn+turn : LET turn=0: RETURN
920 IF a$="n" OR a$="N" THEN RETURN
930 GO TO 900
```

CUBUL MULTICOLOR. Un cub are două dimensiuni? Imposibil desigur, dar acest joc care se joacă pe ecranul monitorului dvs. seamănă foarte mult cu celebrul cub al lui Rubik. Pe un tablou 5×5 calculatorul desenează 25 de pătrăţele colorate dispuse aleatoriu. Liniile orizontale sînt numerotate de la 1 la 5 iar cele verticale sînt numerotate cu 6, 7, 8, 9 şi 0. Calculatorul vă cere să introduceţi numărul unei linii precum şi numărul pentru care doriţi rotirea pătrăţelelor din linia respectivă.

Pe o linie orizontală pătrăţelele se deplasează către dreapta atunci cînd introduceţi un număr pozitiv şi către stînga atunci cînd tastaţi un număr negativ. În aceeaşi manieră, pe verticală ele se deplasează în sus atunci cînd numărul ales este pozitiv şi în jos atunci cînd el este negativ. Cînd un pătrăţel dispare la capătul unui rînd el va reapare în cealaltă parte. Ce urmăreşte acest joc? Trebuie să reaşezaţi pătrăţelele astfel încît fiecare din cele cinci linii orizontale să fie de aceeaşi culoare. Simplu, nu-i aşa? Vă invităm să încercaţi.

```
  5 REM CUB MULTICOLOR
 10 DIM a (6, 6)
 20 FOR x=1 TO 5
 30   FOR y=2 TO 6
 40   LET a (x, y)=y-1
 50   NEXT y
 60 NEXT x
 70 FOR i=1 TO 20
 80   LET k=1+INT (10*RND)
 90   GO SUB 400
120 NEXT i
125 CLS
130 REM DEBUT JOC
140 FOR y=2 TO 6
150   LET p=(y-2)*4-1
160   PRINT AT (p+2), 5; y-1
170   FOR i=1 TO 4
180   PRINT AT (p+i), 7; INK a(1, y); "■" ; INK a(2, y) "■" ;
      INK a(3, y); "■"; INK a(4, y); "■" ; INK a(5, y); "■"
190   NEXT i
200 NEXT y
210 PRINT AT 21, 7; "6 7 8 9 0"
220 INPUT AT 0, 0; "NUMĂR COLOANĂ (0 . 9): " ; k;
    AT 1, 0; "NUMĂR PAŞI (-4 . . 4) : " ; s
225 IF k=0 THEN LET k=10
230 IF (k<1 OR k>10) OR (s<-4 OR s>4) THEN GO TO 220
240 IF s<0 THEN LET s=s+5
260 FOR i=1 TO s
270   GO SUB 400
280 NEXT i
290 GO TO 130
400 IF k>5 THEN GO TO 470
420 FOR x=6 TO 2 STEP -1
430   LET a(x, k+1)=a(x-1, k+1)
440 NEXT x
450 LET a(1, k+1)=a(6, k+1)
460 RETURN
470 FOR y=2 TO 6
480   LET a(k-5, y-1)=a(k-5, y)
490 NEXT y
500 LET a(k-5, 6)=a(k-5, 1)
510 RETURN
```

TO BE OR NOT TO BE. Acest joc asociat lui Shakespeare combinat cu şahul constituie un "pêle-mêle" incitant. Literele sînt dispuse pe o tablă de şah de maniera următoare:

Tabelul 19.1

T	O		B	E		O	R
N	O	T		T	O		B
E		T	H	A	T		I
S		T	H	E		Q	U
E	S	T	I	O	N	.	
W	I	L	L	I	A	M	
S	H	A	K	E	S	P	E
A	R	E		1	6	0	3

Un călăreţ se deplasează pe tablă după mişcarea calului dintr-o partidă de şah. Sărind dintr-o căsuţă într-alta, literele sau simbolurile din căsuţe sînt permutate bulversînd astfel textul de pe tablă.

Calculatorul vă solicită gradul de dificultate (1–100). Introduceţi numărul de mişcări pe care doriţi să le realizeze călăreţul. După fiecare mişcare a calului vi se prezintă imaginea tablei. Sarcina dumneavoastră constă în a ... descurca textul redeplasînd călăreţul pe tablă. Pentru a putea deplasa călăreţul introduceţi un număr de la 1 la 8.

```
  5 REM A FI SAU A NU FI...
 10 DIM x(8) : DIM y(8)
 20 RANDOMIZE
 30 GO TO 4000
 40 REM INIŢIALIZARE
 50 LET X0=4 : LET y0=4
 60 LET b$="TO BE ORNOT TO BE THAT IS THE QUESTION,
    WILLIAM SHAKESPEARE 1603"
 70 LET a$=b$
 80 LET m$ = "1 2 8 3 k 7 4 6 5"
 90 FOR k=1 TO 8
100    READ x(k), y(k)
110 NEXT k
120 RETURN
500 REM MIŞCARE
510 IF begin < > 1 THEN RETURN
520 LET m=INT (RND * 8+1)
530 LET xs=x(m) : LET ys=y(m)
540 IF x0+xs>8 OR x0+xs<1 OR y0+ys>8
    OR y0+ys<1 THEN GO TO 510
550 REM SCHIMBARE LITERE
560 LET P=x0+xs+(y0+ys-1) * 8
570 LET v=x0+(y0-1) * 8
580 LET w$=b$(p)
```

```
590 LET b$(p)=b$(v)
600 LET b$(v)=w$
610 LET x0=x0+xs: LET y0=y0+ys
620 RETURN
1000 REM AFIŞARE TEXT
1010 FOR k=1 TO 8: FOR j=1 TO 8
1020 PRINT AT 1+2 * K, 14+2 * J; b$ ((k-1) * 8+j) ;
1030 NEXT j : NEXT k
1040 RETURN
1500 REM
1510 PRINT AT 1+2 * y0, 14+2 * x0; FLASH 1; OVER 1; " " ;
1520 RETURN
2000 REM AFIŞARE TABLĂ
2010 FOR i=1 TO 17 STEP 2
2020     PLOT 8 * i+116, 155
2030     DRAW 0, -128
2040     PLOT 124, 163-8 * i
2050     DRAW 128, 0
2060 NEXT i
2070 PRINT AT 0, 20; "TABLA"
2080 RETURN
2500 REM
2510 FOR i=1 TO 11 STEP 2
2520     PLOT 8 * i-4, 139
2530     DRAW 0, -80
2540     PLOT 4, 147-8 * i
2545     DRAW 80, 0
2550 NEXT i
2560 FOR k=1 TO 5: FOR j=1 TO 5
2570 PRINT AT 3+2 * k, 2 * j-1; m$ ((k-1) * 5+j);
2580 NEXT j: NEXT k
2590 PRINT AT 1, 0; "MUTĂRI"
2600 RETURN
4000 REM PROGRAM PRINCIPAL
4010 GO SUB 40
4020 INPUT "NIVEL 1-100: " ; : IF l<1 OR l>100 THEN GO TO 4020
4030 LET begin=1
4040 FOR k=1 TO l : GO SUB 500; NEXT k
4050 GO SUB 2000: GO SUB 1000: GO SUB 2500
4060 PRINT AT 20, 20; "NIVEL="; l;
4070 LET move=0 : LET begin=0
4080 GO SUB 1500
4090 INPUT "NIVEL: " ; m
4100 IF m<1 OR m>8 THEN GO TO 4090
4110 LET move=move+1: GO SUB 530
4120 GO SUB 1000: IF b$ <> a$ THEN GO TO 4080
4130 PRINT AT 21, 0; "AŢI REUŞIT ÎN" ; move; "MUTĂRI";
4140 DATA -1, -2, 1, -2,2, -1,2, 1, 1, 2, -1,2, -2,1, -2, -1
```

Jocuri pe AMSTRAD

V-aţi întors din oraş obosit şi fără chef. Găsiţi acasă AMSTRAD-ul plictisit şi descurajat. De cînd nu v-aţi mai jucat împreună? Vă propunem 16 jocuri incitante din care nu vor lipsi cărţile de joc, zarurile, tenis-ul, fantomele, alergările etc. Sperăm să nu vă dezamăgiţi partenerul de întrecere.

1)
```
10 REM zaruri
20 MODE 0
30 PEN 1
```

```
40 BORDER 1
50 PAPER 0
60 s$=SPACE$ (18)
70 r$=CHR$ (129)
80 t$=CHR$ (32)
90 :
100 REM
110 a$=r$+t$+t$
120 b$=t$+r$+t$
130 c$=t$+t$+r$
140 d$=r$+t$+r$
150 e$=t$+t$+t$
160 :
170 LOCATE 2,1
200 DIM dc$ (3, 3)
210 :
220 WHILE k$ < > "s" AND k$ < > "S"
270 : k$=INKEY$ : IF k$=" " THEN 270
280 : LOCATE 1, 12
290 : PRINT s$
300 : PRINT s$
310 : PRINT s$
320 :
330 : FOR n=1 TO 3
340 :     vl%=1+6 * RND (1)
350 :     ON vl% GOSUB 520, 580, 640, 700, 760, 820, 820
360 : NEXT
370 :
380 : LOCATE 1, 12
390 : FOR n=1 TO 3
400 :     PRINT e$; dc$ (1, n); dc$ (2, n); e$; dc$ (3, n)
410 : NEXT
420 :
430 :     FOR k=0 TO 200 : NEXT
440 WEND
450 MODE 1
460 END
470 :
480 :
490 REM
500 :
510 REM
520 dc$ (n, 1)=e$
530 dc$ (n, 2)=b$
540 dc$ (n, 3)=e$
550 RETURN
560 :
570 REM
580 dc$ (n, 1)=a$
590 dc$ (n, 2)=e$
600 dc$ (n, 3)=c$
610 RETURN
620 :
630 REM
640 dc$ (n, 1)=a$
650 dc$ (n, 2)=b$
660 dc$ (n, 3)=c$
670 RETURN
680 :
690 REM
700 dc$ (n, 1)=d$
710 dc$ (n, 2)=e$
720 dc$ (n, 3)=d$
730 RETURN
```

```
740 :
750 REM
760 dc$ (n, 1)=d$
770 dc$ (n, 2)=b$
780 dc$ (n, 3)=d$
790 RETURN
800 :
810 REM
820 dc$ (n, 1)=d$
830 dc$ (n, 2)=d$
840 dc$ (n, 3)=d$
850 RETURN
```

2)

```
10 REM Joc cu mingea (1)
20 MODE 1
30 DEFINT X, Y, D
40 PAPER 2 : INK 2,2
50 PEN 1: INK 1,24
60 BORDER 6
70 CLS
80 S$=SPACE$ (1)
90 DX=1
100 DY=1
110 X=3+35 * RND (1)
120 Y=3+20 * RND (1)
130 :
140 WHILE LEN (a$)=0
150 : LOCATE X, Y
160 : PRINT CHR$ (181)
170 : IF X=1 OR X=40 THEN GOSUB 330
180 : IF Y=1 OR Y=25 THEN GOSUB 380
190 : LOCATE X, Y
200 : PRINT S$
210 : IF X=40 AND Y=25 THEN LOCATE 40, 23: PRINT s$
220 : REM
230 : X=X+DX
240 : Y=Y+DY
250 : a$=INKEY$
260 WEND
270 DEFREAL X, Y, D
280 CLS
290 END
300 :
310 :
320 REM
330 DX=-DX
340 SOUND 1, 0, 5, 5, 0, 0, 5
350 RETURN
360 :
370 REM
380 DY=-DY
390 SOUND 1, 0, 5, 5, 0, 0, 5
400 RETURN
```

3)

```
10 REM Joc cu mingea (2)
20 MODE 1
30 DEFINT X, Y, D
40 PAPER 0 : INK 0,2 : CLS
50 FOR X=624 TO 639
60 : PLOT X, 0
70 : DRAWR 0,399
80 NEXT
90 :
100 GOSUB 490
```

```
110 PEN 1 : INK 1, 24
120 BORDER 24
130 S$=SPACE$ (1)
140 DX=1
150 DY=1
160 X=3+25 * RND (1)
170 Y=3+20 * RND (1)
180 :
190 WHILE LEN (a$)=0
200 : LOCATE X, Y
210 : PRINT CHR$ (231)
220 : IF X=29 OR X=31 THEN GOSUB 570 ELSE flag=0
230 : IF X=1 OR X=39 OR flag=1 THEN GOSUB 390
240 : IF Y=1 OR Y=25 THEN GOSUB 440
250 : LOCATE X, Y
260 : PRINT S$
270 : IF X=40 AND Y=25 THEN LOCATE 40, 23 : PRINT S$
280 : REM
290 : X=X+DX
300 : Y=Y+DY
310 : a$=INKEY$
320 WEND
330 DEFREAL X, Y, D
340 CLS
350 END
360 :
370 :
380 REM
390 DX=-DX
400 SOUND 1, 0, 2+20 * flag, 5, 0, 0, 5
410 RETURN
420 :
430 REM
440 DY=-DY
450 SOUND 1, 0, 2+20 * flag, 5, 0, 0, 5
460 RETURN
470 :
480 REM
490 PEN 2 : INK 2,6
500 FOR Y=1 TO 25
510 : LOCATE 30, Y
520 : PRINT CHR$ (143)
530 NEXT
540 RETURN
550 :
560 REM
570 flag=0
580 IF X=29 AND DX=-1 THEN RETURN
590 IF X=31 AND DX=1 THEN RETURN
600 IF TEST (472, (26-Y) * 16-8)=2 THEN GOSUB 660
610 IF Y=0 OR Y=25 THEN RETURN
620 IF TEST (472, (26-Y-DY) * 16-8)=2 THEN GOSUB 720
630 RETURN
640 :
650 REM
660 flag=1
670 LOCATE 30, Y
680 PRINT S$
690 RETURN
700 :
710 REM
720 IF flag=1 THEN RETURN
730 flag=1
```

```
740 LOCATE 30, Y+DY
750 PRINT S$
760 DY=-DY
770 RETURN
```

4)
```
10 REM Jimmy
20 MODE 0
30 BORDER 15
40 PAPER 0
50 INK 0,1
60 CLS
70 PEN 1
80 INK 1, 24
90 LOCATE 5,4
110 :
130 SYMBOL AFTER 240
140 :
160 SYMBOL 240, 24, 60, 90, 126, 126, 102, 62, 24
170 :
190 SYMBOL 241, 24, 255, 255, 255, 231, 126, 102, 126
200 :
220 SYMBOL 242, 102, 60, 60, 126, 231, 231, 231
230 :
250 SYMBOL 243, 0, 128, 192, 224, 112, 48, 48, 48
260 :
280 SYMBOL 244, 48, 48, 48, 0, 0, 0, 0, 0
290 :
310 SYMBOL 245, 0, 1, 3, 7, 14, 12, 12, 12
320 :
340 SYMBOL 246, 12, 12, 12, 0, 0, 0, 0, 0
350 :
370 SYMBOL 247, 0, 0, 0, 48, 48, 48, 48, 48
380 :
400 SYMBOL 248, 112, 240, 192, 128, 0, 0, 0, 0
410 :
430 SYMBOL 249, 231, 231, 231, 231, 231, 231, 231, 231
440 :
460 SYMBOL 250, 0, 0, 0, 0, 0, 0, 192, 192
470 :
490 SYMBOL 251, 0, 0, 0, 0, 0, 0, 3, 3
500 :
520 SYMBOL 252, 0, 0, 0, 36, 60, 60, 255, 255
530 :
550 SYMBOL 253, 0, 24, 60, 126, 126, 60, 24, 0
560 :
580 SYMBOL 254, 60, 126, 255, 126, 60, 24, 24, 24
590 :
600 :
620 :
640 :
660 :
670 :
680 PEN 2 : INK 2, 16
690 LOCATE 11, 10
700 PRINT CHR$ (240)
710 :
720 PEN 3 : INK 3, 6
730 LOCATE 10, 11
740 PRINT CHR$(245); CHR$(241); CHR$(243)
750 :
760 LOCATE 10, 12
770 PRINT CHR$(246); CHR$(32); CHR$(244)
780 PEN 4 : INK 4, 21
```

```
790 LOCATE 11, 12
800 PRINT CHR$ (242)
810 :
820 PEN 5 : INK 5,0
830 LOCATE 10, 13
840 PRINT CHR$(251); CHR$(249); CHR$(250)
850 :
870 PEN 3
880 WHILE LEN (a$)=0
890 : LOCATE 12, 11
900 : PRINT CHR$ (243)
910 : LOCATE 12,12
920 : PRINT CHR$ (244)
930 : FOR k=0 TO 200 : NEXT
940 : LOCATE 12, 12
950 : PRINT CHR$ (32)
960 : LOCATE 12, 10
970 : PRINT CHR$ (247)
980 : LOCATE 12, 11
990 : PRINT CHR$ (248)
1000 : FOR k=0 TO 200 : NEXT
1010 : LOCATE 12, 10
1020 : PRINT CHR$ (32)
1030 : a$=INKEY$
1040 WEND
1050 PEN 1
1060 MODE 1
1070 END
```

5)

```
10 REM Alergări
20 PAPER 0 : INK 0,26
30 PEN 1 : INK 1,3
40 BORDER 6
50 MODE 0
60 :
70 SYMBOL 240, 0, 0, 0, 0, 1, 1, 1, 0
80 SYMBOL 241, 6, 14, 15, 14, 248, 112, 127, 240
90 SYMBOL 242, 1, 15, 48, 32, 32, 0, 0, 0
100 SYMBOL 243, 252, 4, 7, 0, 0, 0, 0, 0
110 SYMBOL 244, 0, 0, 0, 0, 0, 0, 0, 1
120 SYMBOL 245, 0, 0, 0, 0, 6, 14, 207, 238
130 SYMBOL 246, 1, 0, 7, 7, 118, 30, 1, 1
140 SYMBOL 247, 120, 240, 240, 220, 0, 0, 0, 192
150 :
160 rn1$=CHR$ (240)+CHR$ (241)
170 rn2$=CHR$ (242)+CHR$ (243)
180 rn3$=CHR$ (244)+CHR$ (245)
190 rn4$=CHR$ (246)+CHR$ (247)
200 :
210 WHILE LEN (a$)=0
220 : FOR X=1 TO 19 STEP 2
230 :   LOCATE X, 15
240 :   PRINT rn1$
250 :   LOCATE X, 16
260 :   PRINT rn2$
270 :   FOR n=1 TO 100 : NEXT
280 :   LOCATE X, 15
290 :   PRINT SPACE$ (2)
300 :   LOCATE X, 16
310 :  PRINT SPACE$ (2)
320 :   IF X=19 THEN 420
330 :   LOCATE X+1, 15
340 :   PRINT rn3$
350 :   LOCATE X+1, 16
```

```
360 : PRINT rn4$
370 : FOR n=1 TO 100 : NEXT
380 : LOCATE X+1, 15
390 : PRINT SPACE$ (2)
400 : LOCATE X+1, 16
410 : PRINT SPACE$ (2)
420 : NEXT
430 : a$=INKEY$
440 : FOR n=1 TO 100 : NEXT
450 WEND
460 MODE 1
470 END
```

6)

```
10 REM
20 BORDER 26
30 PAPER 0 : INK 0, 0
40 PEN 1 : INK 1, 24
50 PEN 2 : INK 2, 20
60 MODE 1
70 :
80 WHILE LEN (a$)=0
90 : cl=1
100 : FOR n=1 TO 720
110 :  X=n : IF X>640 THEN X=640
120 :  R=n-80 : IF R<0 THEN R=0
130 :  Y=INT (200+15 * SIN (X/6))
140 :  W=INT (200+15 * SIN (R/6))
150 : IF n/4=INT (n/4) THEN cl=cl+1
160 : IF cl=3 THEN cl=1
170 : PLOT X, Y, cl
180 : PLOT R, W, 0
190 : NEXT
200 : a$=INKEY$ : REM
210 WEND
220 :
230 PEN 1
240 CLS
250 END
```

7)

```
100 REM Mastermind
110 MODE 0
120 DIM colour(3), guess(3), temp(3)
130 SYMBOL 244, 0, 255, 255, 255, 255, 255, 255, 255
140 lne=0
150 INK 1, 6 : INK 2, 18 : INK 3, 24 : INK 4, 2 : INK 5, 8 : INK 6, 10
160 BORDER 16 : PAPER 10 : CLS
170 LOCATE 2, 1 : PRINT CHR$ (244)+" "+CHR$ (244)+" "+
                        CHR$ (244)+" "+CHR$ (244)
180 FOR i=0 TO 3
190  colour (i)=FIX (RND (TIME) * 3)+1
200  temp (i)=colour(i)
210 NEXT i
220 :
230 WHILE pc<>4 AND lne<>20
240  lne=lne+1
250 FOR i=0 TO 3
260  colour(i)=temp(i)
270 NEXT i
280 :
290 PEN 7
300 LOCATE 1, 25 : INPUT "care este combinaţia"; patt$
310 patt$=UPPER$(patt$)
320 LOCATE 1, 25 : PRINT SPACE$ (20);
330 :
```

```
340 FOR i=0 TO 3
350   guess(i)=0
360   IF MID$(patt$, i+1, 1)="R" THEN guess(i)=1
370   IF MID$(patt$, i+1, 1)="G" THEN guess(i)=2
380   IF MID$(patt$, i+1, 1)="Y" THEN guess(i)=3
390   IF MID$(patt$, i+1, 1)="B" THEN guess(i)=4
400   PEN guess(i)
410   LOCATE 2+i * 2, lne+1 : PRINT CHR$(244);
420 NEXT i
430 :
440 pc=0 : cc=0
450 FOR i=0 TO 3
460   IF guess(i)=colour(i) THEN pc=pc+1 : colour(i)=10 : guess(i)=11
470 NEXT i
480 :
490 FOR i=0 TO 3
500   FOR j=0 TO 3
510     IF guess(i)=colour (j) THEN cc=cc+1 : colour (j)=10 : j=3
520   NEXT j
530 NEXT i
540 :
550 IF pc=0 THEN 590
560 PEN 5
570 FOR i =1 TO pc : LOCATE 10+i * 2, lne+1 : PRINT CHR$ (244); : NEXT i
580 :
590 IF cc=0 THEN 630
600 PEN 6
610 FOR i=1 TO cc : LOCATE 10+pc * 2+i * 2, lne+1 :
    PRINT CHR$(244); : NEXT i
620 :
630 WEND
640 PEN 7
650 :
660 IF pc<>4 THEN 740
670 z=TIME
680 WHILE ((TIME-z)/300)<5 : WEND
690 CLS
700 LOCATE 1, 13 : PRINT "FELICITĂRI"
710 PRINT : PRINT : PRINT
720 PRINT "Ai făcut" ; line; "încercări"
730 GOTO 820
740 CLS
750 LOCATE 1, 10 : PRINT "COMBINAŢIE REUŞITĂ"
760 :
770 FOR i=0 TO 3
780   PEN temp (i)
790   LOCATE 10+i * 2, 12 : PRINT CHR$ (244);
800 NEXT i
810 PRINT
820 END
```

8)

```
100 REM Gobble
110 :
120 REM
130 MODE 0
140 BORDER 6 : INK 1, 2 : INK 2, 18 : PAPER 2 : CLS
150 :
160 REM
170 DIM x(2), y(2), dx(2), dy(2), score(2)
180 x(1)=1 : x(2)=640 : y(1)=200 : y(2)=200
190 a=1 : b=2
200 flag=0
210 :
220 REM
```

```
230 WHILE NOT game . over
240  dy(1)=((INKEY(71)=0)-(INKEY(59)=0)) * 2
250  dy(2)=((INKEY(22)=0)-(INKEY(17)=0)) * 2
260  dx(1)=((INKEY(69)=0)-(INKEY(60)=0)) * 4
270  dx(2)=((INKEY(28)=0)-(INKEY(19)=0)) * 4
280  FOR i=a TO b
290   IFdx(i)+dy(i)=0 THEN 310
300   IF TEST(x(i)+dx(i), y(i)+dy(i))< >2 THEN GOSUB 410 ELSE GOSUB 470
310  NEXT i
320 WEND
330 :
340 REM
350 MODE 0 : CLS
360 LOCATE 1, 10 : PRINT "Jucătorul 1 a realizat"; scor(1)
370 LOCATE 1, 14 : PRINT "Jucătorul 2 a realizat"; scor(2)
380 WHILE INKEY$< > " " : WEND
390 END
400 REM
410 PRINT CHR$(7); CHR$(7); CHR$(7)
420 IF i=1 THEN a=2 ELSE b=1
430 flag=flag+1
440 IF flag=2 THEN game . over=-1
450 RETURN
460 REM
470 x(i)=x(i)+dx(i)
480 y(i)=y(i)+dy(i)
490 p=(i=1) * -1+(i=2) * -3
500 DRAW x(i), y(i), p
510 score(i)=score(i)+1
520 RETURN
```

9)

```
100 REM Tenis
110 balls . left=10
120 DIM goals(2)
130 INK 2, 18 : PAPER 2 : CLS
140 SYMBOL 244, 0, 0, 24, 60, 60, 24, 0, 0
150 SYMBOL 245, 24, 24, 24, 24, 24, 24, 24, 24
160 ball$=CHR$(244) : unball$=" "
170 bat$=CHR$(245)
180 unbat$= " "
190 bx=2 : by=19 : sped=5
200 LOCATE bx,11 : PRINT bat$
210 LOCATE by,11 : PRINT bat$
220 player . 1=11 : player . 2=11
230 MODE 0 : CLS
240 WHILE INKEY$=" " : WEND
250 BORDER 6
260 x%=10 : y%=13
270 IF RND(6)>0.5 THEN dx%=-1 ELSE dx%=1
280 IF RND(6)>0.5 THEN dy%=-1 ELSE dy%=1
290 PEN 0
300 WHILE balls.left>0
310  m=m+1
320  IF m<sped THEN GOTO 380
330  LOCATE x%, y% : PRINT unball$;
340  x%=x%+dx% : y%=y%+dy%
350  IF y%<2 OR y%>24 THEN dy%=-dy%
360  LOCATE x%, y% : PRINT ball$;
370  :
380  REM
390  d.player.1=(INKEY(71)=0)-(INKEY(63)=0)
400  d.player.2=(INKEY(30)=0)-(INKEY(22)=0)
410 temp.1=player.1+d. player.1
```

```
420 temp.2=player.2+d. player.2
430 IF temp.1<24 AND temp.1>2 THEN LOCATE bx, player.1 :
    PRINT unbat$ ; : LOCATE bx, temp.1 : PRINT bat$ ; : player.1=temp.1
440 IF temp.2 < 24 AND temp.2 > 2 THEN LOCATE by, player.2 :
    PRINT unbat$ ; : LOCATE by, temp.2 : PRINT bat$ ; : player.2=temp.2
450 IF m<sped THEN GOTO 500 ELSE m=1
460 IF x%=by AND y%=player.2 THEN PRINT CHR$(7): dx%=-dx%
470 IF x%>by THEN k=1: GOSUB 550
480 IF x%=bx AND y%=player.1 THEN PRINT CHR$(7): dx%=-dx%
490 IF x%<bx THEN k=2: GOSUB 550
500 WEND
500 WEND
510 WHILE INKEY$ <> " " ; WEND
520 END
530 REM
540 REM
550  PRINT CHR$ (7)
560  balls.left=balls.left-1
570  goals(k)=goals(k)+1
580 LOCATE(k-1) * 16+1,1: PRINT USING "##"; goals(k)
590 LOCATE x%, y% : PRINT unball$;
600  x%=10
610 RETURN
```

10)

```
100 REM Atac periculos
110 flag=0 : f=0
120 MODE 1
130 SYMBOL 244, 255, 153, 153, 255, 255, 129, 129, 255
140 SYMBOL 245, 24, 126, 231, 219, 219, 231, 126, 0
150 SYMBOL 246, 231, 102, 60, 24, 24, 24, 24, 24
160 SYMBOL 247, 0, 0, 0, 24, 24, 0, 0, 0
170 shell$=CHR$ (247)
180 bomb$=CHR$ (226)
190 d$=CHR$(245) : c$=CHR$(244)
200 c2$=c$+c$
210 c3$=c2$+c$
220 c7$=c3$+c2$+c2$
230 c10$=c7$+c3$
240 c16$=c7$+c7$+c2$
250 c18$=c2$+c16$
260 PEN 1
270 LOCATE 5, 19: PRINT c3$+SPACE$(4)+c2$+SPACE$(2)+
    c3$+SPACE$(3)+c3$+SPACE$(4)+c3$
280 PRINT SPACE$(4)+c3$+SPACE$(4)+c2$+SPACE$(2)+
    c3$+SPACE$(3)+c3$+SPACE$(4)+c3$
290 PRINT SPACE$(4)+c3$+" "+c10$+SPACE$(3)+c18$
300 PRINT c7$+ " " +c10$+SPACE$(3)+c18$
310 PRINT c7$+ " " +c10$+SPACE$(3)+c18$
320 PRINT SPACE$(4)+c3$+ " " c10$+SPACE$(3) +c18$
330 PRINT SPACE$(4)+c3$+" "+c10$+SPACE$(3)+c18$
340 :
350 a=20
360 b=INT (RND(6) * 38)+1
370 z=1
380 MOVE 0,0
390 LOCATE 25, 1: PRINT "Lovituri reușite" ; : hit=0
400 LOCATE 1,1 : PRINT "Lovituri nereușite" ; : blast=0
410 r=a : s=18
420 :
430 WHILE hit < 555
440  LOCATE 35,1 : PRINT hit;
450  fire=(INKEY(9)=0)
460  left=(INKEY(8)=0)
470  right=(INKEY(1)=0)
```

```
480  IF fire AND flag=0 THEN flag=1 : r=a : PRINT CHR$(7);
490  IF flag=1 THEN LOCATE r, s-1 : PRINT shell$ ; :
     LOCATE r, s : PRINT " " ; : GOSUB 640 : s=s-1
500  k=left-right+(a=39)-(a=1)
510  LOCATE a, 18 : PRINT " ";
520  LOCATE a+k, 18: PRINT d$;
530  a=a+k
540  g=620 * b/40 : h=400 * (25-z-1)/25
550  q=(TEST(g, h)=1)
560  IF f=1 THEN LOCATE b, z: PRINT " " ; : LOCATE b, z+1 :
     PRINT bomb$ ; : z=z+1
570  f=(f+1) MOD 3
580  IF z=24 OR q THEN LOCATE b, z: PRINT " " ; : LOCATE b, z+1 :
     PRINT " "; : z=2 : b=INT (RND(6) * 40)+1 : hit=hit+1
590 WEND
600 END
610 :
620 :
630 REM
640 t=(b=r) AND (s=z+1)
650 IF t THEN SOUND 1, 200, 10, 7, 0, 0, 1 : blast=blast+1 : LOCATE 14, 2 :
    PRINT blast ; : z=2 : b=INT (RND(6) * 40)+1
660 IF t OR s=3 THEN flag=0: LOCATE r, s-1 : PRINT " " :
    LOCATE r, s : PRINT " " : PRINT " " ; : r=a; s=19
670 RETURN
```

11)

```
100 REM Fantome
110 MODE 1 : w=40 : BORDER 6
120 LOCATE 2, 10 : INPUT "Grad de dificultate (1-5)"; sped : sped=6 - sped
130 CLS
140 count=0
150 dead=0
160 m=FIX (RND(TIME) * 5)+1 : p=FIX(RND (TIME) * 5)+3
170 DIM mons (m, 2), man (2)
180 el$=" "
190 :
200 SYMBOL 244, 31, 124, 200, 248, 248, 120, 60, 31
210 SYMBOL 245, 60, 126, 255, 254, 254, 62, 30, 28
220 SYMBOL 246, 28, 28, 8, 127, 28, 20, 20, 54
230 :
240 FOR i=1 TO m
250   mans(i, 1)=FIX(RND(TIME) * w)+1
260   mons(i, 2)=FIX(RND(TIME) * 25)+1
270   PEN 1
280   LOCATE mons(i, 1), mons(i, 2): PRINT CHR$(244);
290 NEXT i
300 :
310 FOR i=1 TO p
320   x=FIX (RND (TIME) * w)+1
330   y=FIX (RND (TIME) * 25)+1
340   PEN 3
350   LOCATE x, y : PRINT CHR$(245);
360 NEXT i
370 :
380 man(1)=FIX(RND(TIME) * w)+1
390 man(2)=FIX(RND(TIME) * 25)+1
400 dx=0 : dy=0
410 PEN 2
420 LOCATE man(1), man(2) : PRINT CHR$(246);
430 WHILE INKEY$=" " : WEND
440 :
450 WHILE m > 0 AND NOT dead
```

```
460   PEN 2
470   LOCATE man(1), man(2): PRINT CHR$(246);
480   dx=(INKEY(8)=0)-(INKEY(1)=0)
490   dy=(INKEY(0)=0)-(INKEY(2)=0)
500   cx=man(1)+dx: cy=man(2)+dy: GOSUB 710
510   IF man(1)+dx=40 OR man(1)+dx=1 THEN dx=0
520   IF man(2)+dy=25 OR man(2)+dy=1 THEN dy=0
530   IF c=1 THEN GOSUB 750
540   IF c=3 THEN el$=CHR$(245) ELSE el$= " "
550   IF dx=0 AND dy=0 THEN GOTO 620
560   IF el$=" " THEN PEN 0 ELSE PEN 3
570   LOCATE man(1), man(2) : PRINT el$
580   PEN 2
590   man(1)=man(1)+dx
600   man(2)=man(2)+dy
610   LOCATE man(1), man(2) : PRINT CHR$ (246);
620   count=(count+1) MOD sped
630   IF count=0 THEN GOSUB 840
640 WEND
650 :
660 WHILE INKEY$ <> " " : WEND
670 IF m=0 THEN INK 0, 5, 8
680 END
690 :
700 REM
710 c=TEST ((cx-1) * 16+4, (25-cy) * 16+4)
720 RETURN
730 :
740 REM
750 INK 0, 9, 4
760 FOR z=20 TO 0 STEP - 1
770   SOUND 1, 10+n, 100, 7
780   WHILE SQ(1) > 127 : WEND
790 NEXT z
800 dead=-1
810 RETURN
820 :
830 REM
840 FOR i=1 TO m
850   LOCATE mons(i, 1), mons(i,2) : PRINT " " ;
860   x=SGN(man(1)-mons(i,1))
870   y=SGN(man(2)-mons(i,2))
880   cx=mons(i,1)+x : cy=mons(i,2)+y : GOSUB 710
890 LOCATE 1,1 : PRINT c
900   IF c=3 THEN GOSUB 990 : GOTO 950
910   PEN 1
920   IF c=1 THEN LOCATE cx, cy : PRINT CHR$(244); :
930   mons(i,1)=cx : mons(i,2)=cy
940   LOCATE cx, cy : PRINT CHR$(244);
950 NEXT i
960 RETURN
970 :
980 REM
990 PRINT CHR$(7);
1000 LOCATE mons(i,1), mons(i,2) : PRINT " " ;
1010 IF i=m THEN m=m -1 : RETURN
1020 FOR j=i TO m-1
1030   mons(j,1)=mons(j+1,1)
1040   mons(j,2)=mons(j+1,2)
1050 NEXT j
1060 m=m-1
1070 RETURN
```

12)

```
100 REM Ruletă
110 MODE 1
120 BORDER 9
130 INK 0,21 : INK 1,8
260 A$=INKEY$ : IF A$="" THEN 260
270 REM
280 :
290 MODE 2
300 ORIGIN 320, 199
310 FOR I=1 TO 30
320   MOVE 0,0
330   K=2 * PI/30 * I
340   DRAW 150 * COS(K), 150 * SIN(K)
350 NEXT I
360 PEN 1
370 FOR N=0 TO 29
380   K=(2 * N+1)/30 * PI
390   LOCATE (155 * COS(K)+320)/8,25-(155 * SIN(K)+199)/16
400   PRINT N+1;
410 NEXT N
420 :
440 A$=INKEY$ : IF A$="" THEN 440
450 LOCATE 1,25 : PRINT
460 LOCATE 1,25 : PRINT
460 REM
470 T=TIME
480 T1=(INT(RND(2) * 20)) * 300+100
490 EVERY 15,1 GOSUB 660
500 p=10
510 WHILE TIME < T+T1
520   A=(TIME-T)/600 * PI
530   MOVE 0,0
540   PRINT CHR$(23)+CHR$(1)
550   DRAW 150 * COS(A), 150 * SIN(A)
560   MOVE 0,0
570   DRAW 150 * COS(A), 150 * SIN(A)
580   FOR z=1 TO p : NEXT z
590   p=p+1
600 WEND
610 PRINT CHR$(23)+CHR$(0)
620 MOVE 0,0 : DRAW 170 * COS(A), 170 * SIN(A)
630 MOVE 0,0 : DRAW 170 * COS(A+0.01), 170 * SIN(A+0.01)
640 LOCATE 1,1
650 END
660 PRINT CHR$(7);
670 RETURN
```

13)

```
100 REM Joc de cărți
110 CLS : INK 1, 3 : INK 2,0 : INK 0,9
120 SYMBOL 244, 54, 127, 127, 127, 62, 28, 8, 0
130 SYMBOL 245, 8, 28, 28, 107, 127, 107, 8, 28
140 SYMBOL 246, 8, 28, 62, 127, 62, 28, 8, 0
150 SYMBOL 247, 8, 28, 62, 127, 127, 127, 28, 62
160 h$=CHR$(244)
170 c$=CHR$(245)
180 d$=CHR$(246)
190 s$=CHR$(247)
200 pack$=""
210 heart$=""
220 club$=""
230 diamond$=""
250 hand1$=""
260 hand2$=""
```

```
270 hand3$=""
280 hand4$=""
290 suit$=heart$ : x$=h$
300 GOSUB 680 : heart$=suit$
310 suit$=club$ : x$=c$
320 GOSUB 680 : club$=suit$
330 suit$=diamond$ : x$=d$
340 GOSUB 680 : diamond$=suit$
450 suit$=spade$ : x$=s$
360 GOSUB 680 : spade$=suit$
370 pack$=heart$+club$+diamond$+spade$
380 :
390 :
400 LOCATE 15, 10 : PRINT "Joc de cărți"
410 shuffled$=""
420 FOR i=1 TO 50
430   PRINT " . " ;
440   p=INT(RND(10) * (53-i)+1) * 2-1
450   shuffled$=shuffled$+MID$(pack$,p,2)
460   pack$=LEFT$(pack$, p-1)+MID$(pack$, p+2)
470 NEXT i
480 shuffled$=shuffled$+pack$
490 FOR i=1 TO 13
500   hand1$=hand1$+LEFT$(shuffled$,2)
510   shuffled$=MID$(shuffled$,3)
520   hand2$=hand2$+LEFT$(shuffled$,2)
530   shuffled$=MID$(shuffled$,3)
540   hand3$=hand3$+LEFT$(shuffled$,2)
550   shuffled$=MID$(shuffled$,3)
560   hand4$=hand4$+LEFT$(shuffled$,2)
570   shuffled$=MID$(shuffled$,3)
580 NEXT i
590 DIM x(4)
600 CLS
610 hand$=hand1$ : x=14 : y=1 : GOSUB 760
620 hand$=hand2$ : x=5 : y=8 : GOSUB 760
630 hand$=hand3$ : x=14 : y=16 : GOSUB 760
640 hand$=hand4$ : x=28 : y=8 : GOSUB 760
650 LOCATE 1,23
660 END
670 :
680 REM
690 FOR i=2 TO 9
700   suit$=suit$+RIGHT$ (STR$(i), 1)+x$
710 NEXT i
720 suit$="A"+x$+suit$+"T"+x$+"J"+x$+"Q"+x$+"K"
730 RETURN
740 :
750 :
760 REM
770 x(1)=x : x(2)=x : x(3)=x : x(4)=x
780 FOR i=1 TO 13
790 card$=MID$ (hand$, i * 2-1,2)
800 IF RIGHT$(card$,1)=h$ OR RIGHT$(card$, 1)=d$
    THEN PEN 1 ELSE PEN 2
810 z=ASC(RIGHT(card$,k)) - 243
820 LOCATE x(z), y+z : PRINT card$;
830 x(z)=x(z)+2
840 NEXT i
850 RETURN
```

14)
```
100 REM Fantome
110 ENV 1, 5, 3, 4, 2,–4, 4, 3, 2, 4, 1, 0, 12, 2,–3, 4
120 DIM x(10), y(10)
130 DEF FNrand(d)=FIX (RND(TIME) * d)+1
140 MODE 0 : PAPER 2 : CLS : BORDER 6
150 SYMBOL 244, 24, 60, 90, 126, 36, 90, 66, 129
160 PEN 1
170 :
180 WHILE 1=1
190 count=FIX (RND(TIME) * 9)+1
200 INK 2,2 : CLS
210 x(1)=FNrand(20)
220 y(1)=FNrand(22)
230 :
240 FOR i=2 TO count
250   x(i)=FNrand(20)
260   y(i)=FNrand(22)
270   FOR j=1 TO i–1
280     IF x(i)=x(j) AND y(i)=y(j) THEN i=i–1
290   NEXT j
300 NEXT i
310 :
320 FOR i=1 TO count
330   LOCATE x(i), y(i) : PRINT CHR$ (244);
340 NEXT i
350 :
360 flag=0
370 WHILE flag=0
380   flag=–1
390   LOCATE 1,24 : PRINT "Număr fantome"
400   ans$=INKEY$ : IF ans$=" " THEN 400
410   LOCATE 1, 24 : PRINT SPACE$ (30);
420   ans=VAL (ans$)
430   IF ans=count THEN INK 2, 5, 8 : GOSUB 470
      ELSE SOUND 1, 200, 10, 0, 1, 0, 4 : flag=0
440 WEND
450 WEND
460 END
470 tempo=2·5
480   RESTORE 570
490   FOR x=1 TO 20
500   READ pitch, duration
510   pitch=pitch–140
520   freq=440 * (2∧((pitch–10/12))
530   pitchnum=ROUND (125000/freq)
540   SOUND 1, pitchnum, duratian * tempo, 15, 0, 0, 0
550   NEXT x
560 RETURN
570 DATA 109, 6, 117, 2, 121, 6, 129, 2, 137, 8, 157, 6, 165, 2, 169, 8, 165, 6,
        157, 8, 137, 8
580 DATA 109, 6, 117, 2, 121, 6, 129, 2, 137, 8, 157, 4, 110, 2, 137, 2, 145, 8
```

15)
```
100 REM
110 MODE 0
120 ORIGIN 0, 0
130 FOR i=1 TO 200
140   c=i MOD 16
150   b1=i MOD 31 : b2=FIX (RND(TIME) * 25)+1
160   BORDER b1,b2
170   MOVE i, i
180   DRAWR 0, 400–2 * i, c : DRAWR 640–2 * i, 0, c
```

```
190    DRAWR 0, 2 * i–400, c : DRAWR i * 2–640, 0, c
200    SOUND 1, i, 10, 15, 0, 0, 0
210    SOUND 2, 200–i, 10, 15, 0, 0, 0
220    SOUND 3, FIX(RND(TIME) * 200)+1, 10, 15, 0, 0, 0
230 NEXT i
240 CALL 0
```

16)
```
100 REM
110 MODE 0
120 ORIGIN 320, 200
125 c=1
130 WHILE 1=1
140    dx=(INKEY (8)=0)–(INKEY (1)=0)
150    dy=(INKEY (2)=0)–(INKEY (0)=0)
170    DRAWR dx, dy
180 WEND
```

Jocuri pe COMMODORE

NIM. Să presupunem că un prieten vă provoacă spunîndu-vă: „Hai să ne jucăm. Am aici 21 de obiecte din care poţi lua de fiecare dată unul, două sau trei. Cine rămîne cu ultimul obiect pierde. Deoarece sînt un bun jucător, te las să începi tu primul!"

Bine, vă decideţi într-un tîrziu şi încercaţi. Pierdeţi. Veţi încerca din nou şi veţi pierde, mereu, mereu pînă cînd vă veţi plictisi. Acesta este Nim. Dacă ştiţi secretul nu puteţi pierde. Cheia este să . . .

```
120 REM COMPUTER RESPONSES
130 W$(1)="I THINK I'LL TAKE"
140 W$(2)="THIS TIME I WANT"
150 W$(3)="I HAVE TAKEN"
160 W$(4)="I'LL HAVE"
170 W$(5)="I TOOK"
180 W$(6)="LET ME HAVE"
190 W$(7)="THIS TIME GIVE ME"
200 S=21
210 PRINT"{CLR}LET'S PLAY THE GAME OF"
220 PRINT"NIM. {2 SPACES}WE START WITH 21"
230 PRINT"THINGS. {2 SPACES}EACH TURN WE"
240 PRINT"CAN TAKE AWAY ONE, TWO,"
250 PRINT"OR THREE THINGS."
260 PRINT
270 PRINT"THE ONE WHO HAS TO"
280 PRINT"TAKE THE LAST THING"
290 PRINT"LOSES THE GAME."
300 PRINT
310 PRINT"YOU GET TO GO FIRST!"
320 PRINT"HOW MANY DO YOU WANT?"
330 REM CLEAR KEYBOARD BUFFER
340 POKE 198,0
350 REM GET NUMBER & MAKE SURE
360 REM IT IS 1, 2, OR 3
370 GET N$
380 IF N$=" " THEN 370
390 N=ASC(N$) – 48
400 IF N<1 OR N>3 THEN PRINT"NO! YOU CAN TAKE ONLY 1,2, OR 3!" :
    GOTO 370
410 REM CHECK FOR DECIMAL NUMBERS
420 IF N <> INT(N) THEN PRINT "NO! USE ONLY 1, 2, {SPACE}OR 3!"
430 PRINT
```

```
440 PRINT"YOU TOOK"; N
450 PRINT S; "–"; N"="; S–N
460 S=S–N
470 REM THINKING LOOP
480 RN=INT(RND(1) * 3500)+300
490 FOR J=1 TO RN
500 NEXT J
510 IF N=1 THEN R=3
520 IF N=2 THEN R=2
530 IF N=3 THEN R=1
540 PRINT
550 REM GET RESPONSE
560 RN=INT(RND(1) * 7)+1
570 PRINT W$(RN); R
580 PRINT
590 PRINT S; "–"; R; "–"; S–R
600 S=S–R
610 IF S=1 THEN 640
620 PRINT
630 GOTO 320
640 PRINT
650 PRINT"YOU HAVE TO TAKE THE"
660 PRINT"LAST ONE SO YOU LOSE!"
670 PRINT
680 PRINT"TO PLAY AGAIN PRESS"
690 PRINT"THE 'A' KEY."
700 GET A$
710 IF A$=" " THEN 700
720 IF A$ <> "A" THEN 700
730 GOTO 200
```

REACȚIE. V-au obosit jocurile în care singura dvs. sarcină era să manevrați un "mîncător" de puncte pasiv, în jurul ecranului? Dacă da „Reacție" este un joc pentru dvs. cu 20 de nivele de dificultate. Vi se cere să controlați o cărămidă mare albă ce aleargă în jurul ecranului. Sună simplu, dar jocul conține multe trucuri. De fiecare dată cînd cărămida albă lasă în urmă un zid și . . .

```
5 POKE 53281,0 : POKE 53280,12
10 PRINT "{CLR}{WHT}" : POKE 56,48 : PRINT" {8 DOWN}
   {10 RIGHT} ENTERING CHARACTERS" : GOSUB 1000
99 REM LINE 100–200 ARE THE MAIN LOOP***
100 POKE 53272,29 : POKEX+54272,1 : POKEX, A : OM=M : IF 41–P
   EEK(2)= . THENM=OM : GOTO 120
110 M=41–PEEK(2)
120 X=X+M : IFX<CORX>DTHENX=X–M : M= .
130 P=PEEK(X) : IF(P=AANDM<> . )ORP=ETHEN 500
135 POKEG , .
140 IFP= . THENDE=DE+E : POKEG, 250 : IFDE=STHEN 400
150 IFP=FTHEN 300
200 GOTO 100
300 REM HIT PRIZE
400 REM WIN ROUTINE * * *
410 GOSUB 610
420 FORI=160 TO 220 STEP 5 : FORJ=ITOI+30 : POKEG, J : NEXT : N
   EXT : POKEG , . : POKE 53272,21
430 PRINT "{2 DOWN}{WHT}{5 DOWN}{15 RIGHT} YOU WON!"
435 PRINT "{DOWN}{WHT}{3 DOWN}{10 RIGHT} PREPARE FOR LEVEL";
   S+1 : S=S+1
440 FORI=1 TO 3000 : NEXT : POKE 53272,29 : GOSUB 1110 : GOTO 100
500 REM LOSE ROUTINE * *
```

```
510 GOSUB 610
520 FORI=220 TO 127 STEP- . 5 : POKEG, I : NEXT
530 PRINT "{DOWN}{WHT}{6 DOWN}{5 RIGHT} YOU DIDN'T COMPLETE
    LEVEL"; S; "{LEFT}."
535 PRINT "{DOWN}{12 RIGHT} PERCENTAGE"; INT (DE/S * 100); "%"
540 PRINT "{DOWN}{10 RIGHT} PREPARE TO RETURN ..."
550 FORI=1 TO 4000 : NEXT : POKE 53272,29 : GOSUB 1110 : GOTO 100
610 PRINT "{CLR}{WHT}" : POKE 53272,21 : PRINT" {5 DOWN}{15 SPACES}
    REACTION"
620 REM
630 RETURN
1000 REM GAME SET-UP **
1010 POKE 56334, PEEK (56334) AND 254
1015 POKE1, PEEK(1) AND 251
1020 FOR I=1024 TO 1536 : POKEI+12288, PEEK (I+53248) : NEXT
1025 POKE1, PEEK(1) OR 4 : POKE 56334, PEEK (56334) OR 1
1050 X=1482 : A=3 : B=5 : C=1064 : D=2023 : E=1 : F=2
1052 FORI=12288+24 TO 12288+31 : POKEI, 255 : NEXT
1054 DATA0, 60, 36, 36, 36, 60, 0, 0, 0, 60, 60, 60, 60, 60, 0, 0
1056 FOR I=12288 TO 12288+15 : READP : POKEI, P : NEXT
1058 FORI=12288+256 TO 12288+256+7 : POKEI, . : NEXT : GOSUB 1300 : SYS 49152
1060 PRINT "{CLR}" : POKE 53272,29
1070 PRINT "{5 DOWN}{PUR}{RVS}{11 RIGHT} USE THE JOY STICK TO"
1075 PRINT "{2 DOWN}{PUR}{6 RIGHT}{RVS} MOVE AROUND
     {SPACE} THE PLAY- FIELD."
1080 PRINT "{GRN}{2 DOWN}{RVS}{11 RIGHT} HIT THE {OFF} @ {RVS}'S
     WHILE"
1085 PRINT "{2 DOWN}{9 RIGHT}{RVS} TRYING TO AVOID THE
     {OFF}A{RVS}'S."
1090 PRINT "{10 RIGHT}{DOWN}{YEL}{RVS} DIFFICULTY (1-20)"; : INPUTS
1100 IFS<1 ORS>20 THENE=32 : S=S-100
1110 PRINT "{CLR}" : DE= . : OM= . : M= .
1120 FORI=1 TOS : Q=INT (959 * RND(1)+C) : POKEQ+54272,1 : P
     OKEQ, E : NEXT
1125 FORI=1 TOS : Q=INT (595 * RND(1)+C) : POKEQ+54272,1 : P
     OKEQ, . : NEXT : E=1
1130 PRINT "{HOME}{RVS}{GRN} REACTION"
1140 IFPEEK(A)= . THENA=A+1 : GOTO 1140
1200 RETURN
1300 I=49152
1310 READ B : IF B=256 THEN RETURN
1320 POKE I, B : I=I+1 : GOTO 1310
1330 DATA 120, 169, 13, 141, 20, 3, 169
1340 DATA 192, 141, 21, 3, 88, 96, 173
1350 DATA 0, 220, 41, 15, 73, 15, 168
1360 DATA 185, 29, 192, 133, 2, 76, 49
1370 DATA 234, 41, 81, 1, 41, 42, 41
1380 DATA 41, 41, 40, 256
```

VECHIUL VEST. Jocul vă plasează într-un oraș din vechiul Vest și oferă 1 000 $ recompensă dacă-l prindeți viu pe Black Bart. Îl suspectați că se ascunde undeva în oraș dar trebuie să vă folosiți inteligența pentru a-l localiza și a-l captura.

```
100 PRINTCHR$ (147) : C1=0 : K1=0 : L1=0 : B1=0 : W1=0 : J1=0 : S1=0
110 DIM N$(19), L(26), M(10,4), R$(10)
120 GOSUB 500 : L=1 : PRINT "I HEAR A RASPING NOISE" : FORTD=1 TO 1000:
    NEXT TD
125 PRINT "I HEAR A HORSE!" : FOR TD=1 TO 300 : NEXT TD
126 PRINT : FORCY=1 TO 10 : PRINT "{2 SPACES} CLOPPITY" : FORTD=1 TO 200 :
    NEXTTD : NEXTCY
127 FOR TD=1 TO 1000 : NEXT TD : PRINT
```

```
130 GOSUB 700
135 IFM (4,2) < > 0 ORL (13) < > 3 THEN 140
136 PRINT "{CLR}" : PRINT "YOU CAUGHT BLACK BART!"
137 PRINT "THE REWARD IS YOURS!"
138 END
140 PRINT : PRINT "TELL ME WHAT TO DO" :  GOSUB 1000
145 IF S=1 THENS=0 : GOTO 140
150 FOR I=1 TO LEN(V$) STEP 2
160 IF MID$(V$, I, 2)=A$THENV=(I+1)/2 : GOTO 200
170 NEXTI
180 PRINT "I DON'T KNOW HOW TO DO THAT" : GOTO 140
200 IF B$=" "THEN 300,
210 FOR I=1 TOLEN(N$) STEP 2
220 IF MID$ (N$, I, 2)< >B$THEN 230
225 N=(I+1)/2
227 IFN=21 THENN=9
229 GOTO 300
230 NEXTI
300 ON V GOTO 350, 400, 400, 450, 480, 800, 800, 830, 130, 800, 900, 980, 950
309 REM UNLOCK DRAWER {2 SPACES} DESK OR CELL
310 IF N=6 ANDL(17)=0 ANDL=2 THENJ1=1 : M(2, 3)=3 : GOSUB1
    150 : M(3, 4)=2 : GOTO 130
320 PRINT "CAN'T DO THAT YET"
330 GOTO 130
350 IFN< >7 ANDN<22 THEN 380
355 IF N=7 ANDM(4, 2)< >0 AND(L=5 ORL=3) THENL=4 : GOTO 130
360 IFM(L, N-21)=0 THEN 395
370 L=M(L, N-21) : GOTO 130
380 IF N< >4 OR L< >7 THEN 385
381 PRINT "SPLASH- -I'M ALL WET" : L(18)=7
382 IFL(11)=50 THENPRINT "I FOUND SOMETHING" : L(11)=7 : GOTO 130
385 IFN=5 ANDL=9 THENL=10 : GOTO 130
395 PRINT "CAN'T GO THAT WAY" : GOTO 130
400 IFC>4 THENPRINT "GO TOO MUCH- -TAKE INVENTORY" : GOTO 140
405 IFN=14 THEN 425
410 IFL(N)< >LTHENPRINT "I DON'T SEE IT HERE" : GOTO 140
420 IFN<80 ORN>19 THENPRINT "CAN'T TAKE THAT" : GOTO 140
425 IFN=14 ANDL< >2 ANDL(16)< >0 THENPRINT "CAN'T DO THAT YET"
    : GOTO 140
430 IFN=14 ANDL=2 THENN$(3)= "SINK FULL OF WATER" : W1=1 : GOTO 130
435 IFN=14 ANDL(16)=0 THENN$(16)= "BUCKET OF WATER" : W1=1 : GOTO 130
440 IF N=13 AND S1=1 THEN 444
441 IFN< >13 THEN 444
442 PRINT "BLACK BART GETS MAD!" : PRINT "HE DRAWS HIS REVOLVER!"
443 PRINT "THAT'S THE END" : END
444 IFN=13 ANDL(11)< >0 THENPRINT "CAN'T DO THAT YET" : GOTO 140
445 PRINT "OKAY" : C=C+1 : L(N)=0 : GOTO 140
450 IFL1=1 THENPRINT "IT'S NOT LOCKED" : GOTO 140 RUN
460 IFL=2 ANDL(9)=0 ANDL1=0 AND(N=2 ORN=20 ORN=26) THEN 465
462 GOTO 470
465 L1=1 : K1=1 : PRINT "OKAY" : N$(2)= "UNLOCKED DESK"
470 GOTO 140
480 IFN< >2 ANDN< >20 ANDN< >26 THENPRINT "CAN'T" : GOTO 140
490 IFK1=1 ANDL=2 ANDL(17)=50 THENL(17)=2 : GOTO 495
492 PRINT "CAN'T" : GOTO 140
495 PRINT "OKAY- -THERE'S SOMETHING IN THERE" : GOTO 140
500 FORI=1 TO19 : READN$(I) : NEXTI
510 DATA POSTER, DESK WITH LOCKED DRAWER, SINK, HORSE {SPACE}
    TROUGH, ROOM IN BACK
511 DATA JAIL CELL-LOCKED, EMPTY WINDOW, BARS, BOBBY PIN, SACK OF
    CEMENT
512 DATA SHINY STAR, BOTTLE OF ELIXIR, BLACK BART, WATER, WATER, BUCKET,
    KEY
```

```
514 DATA WATER, EMPTY WINDOW
550 VI$="NSEW"
560 V$="GOTAGEPIOPPUDRMILOGISWFIEREUN"
570 N$="PODESITRROJAWIBABOCESTWHBLWAWABUKEWAWIDRPI
    NOSOEAWELO"
600 FORI=1 TO 19 : READL(I) : NEXTI
610 DATA 1, 2, 3, 7, 9, 2, 3, 5, 8, 8, 50, 9, 10, 50, 11, 7, 50, 11, 5
620 FORI=1 TO 10 : FORJ=1 TO 4 : READM(I, J) : NEXTJ : NEXTI
630 DATA 7, 6, 2, , , , , 1, , , , 0, 3, 5, , , 0, , , 6
640 DATA 1, , 5, , , 1, 8, 9, , , , 7, , , 7, , , , 9.
650 FORI=1 TO 10 : READR$(I) : NEXTI
660 DATA DUSTY STREET, OFFICE, JAIL, WINDOW, DUSTY STREET BEHIND JAIL
670 DATA STREET CORNER, DUSTY STREET, GENERAL STORE, SALOON, ROOM
    IN BACK
675 R$(2)="SHERIFF'S"+R$(2)
680 RETURN
700 PRINT "I AM IN A" R$(L)
710 PRINT "I SEE";
720 FOR I=1 TO 19
730 IF L(I)=L THEN PRINTN$(I); " "; : S=1
740 NEXTI
750 IF S=0 THENPRINT "NOTHING SPECIAL"
760 IF S=1 THEN PRINT : S=0
765 IF L=0 THEN RETURN
770 PRINT "OBVIOUS EXITS ARE";
775 FORI=1 TO 4 : IFM(L, I)=0 THEN 785
780 PRINTMID$(V1$, I, 1)+" ";
785 NEXTI : PRINT
790 RETURN
800 IFN=10 ORN=8 THEN 830
805 IFN=12 ANDS1=0 THEN 870
810 IFL(N)=0 THENL(N)=L : C=C-1 : PRINT "OKAY" : GOTO 130
820 PRINT "I'M NOT CARRYING IT" : GOTO 140
830 PRINT "WHERE" : INPUT C$
840 IFLEFT$(C$, 2)< >"SI" ANDLEFT$(C$, 2)< >"WA" ANDLEFT$(C$, 2)< >"BU"
    THEN 845
842 IFW1=1 ANDN=10 ANDL(10)=0 THEN 850
843 GOTO 847
845 IFLEFT$(C$, 2)="WI" ANDL(8)=0 ANDC1=1 ANDL=4 ANDN=8 THEN 860
847 PRINT "CAN'T DO THAT YET" : GOTO 140
850 PRINT "OKAY . {2 SPACES} IT'S MIXED"; C1=1 : L(10)=50 : C=C-1
855 IFLEFT$(C$, 2)="BU" THENN$(16)="BUCKET OF CEMENT MIXTURE" :
    GOTO 130
856 N$(15)="CEMENT MIXTURE" : L(15)=L : GOTO 130
860 PRINT "OKAY" : N$(7)="BARRED WINDOW" : L=3 : M(4, 2)=0 : L(8)=50 :
    C=C-1 : GOTO 130
870 IFL(12)< >0 THEN 847
875 IFA$< >"GI" THEN 900
876 IFL< >10 THEN 847
880 L(12)=50 : PRINT "BLACK BART GULPS{6 SPACES} ELIXIR AND
    IMMEDIATELYFALLS ASLEEP"
890 S1=1 : C=C-1 : N$(13)="SLEEPING"+N$(13) : GOTO 140
900 IFN=12 ANDL(12)=0 THENL(12)=50 : PRINT "GLUG GLUG GLUG" : C=C-1 :
    GOTO 140
910 IFN=12 THENPRINT "CAN'T DO THAT YET" : GOTO 140
920 PRINT "CAN'T DRINK THAT" : GOTO 140
950 IFN< >1 ORL< >THENPRINT "CAN'T" : GOTO 140
960 PRINT "WANTED ALIVE-BLACK BARTI $1000 REWARD"
970 GOTO 140
980 IFN< >3 ORL< >3 THENPRINT "CAN'T" : GOTO 140
990 PRINT "HOW"; : INPUTC$
995 IFLEFT$(C$, 2)="WA" THENL(14)=3 : W1=1
996 GOTO 130
```

```
1000 INPUT A$ : B$=" "
1005 IF LEFT$(A$, 2)="QU" THENEND
1010 IFLEN(A$)=1 THEN 1060
1020 FORI=1 TOLEN(A$)
1030 IFMID$(A$, I, I)=" " AND LEN(A$)>I+1 THENB$=MID$(A$, I+1, 2) :
     GOTO 1045
1040 NEXTI
1045 A$=LEFT$(A$, 2) : RETURN
1060 IFA$ < > "I" THEN 1100
1070 PRINT "I'M CARRYING"; : T=L : L=0 : GOSUB 720
1080 L=T : S=1 : RETURN
1100 FORI=1 TO 4
1110 IFA$<>MID$(V1$, I, 1) THEN 1130
1120 A$="GO" : B$=MID$(N$, 2 * I+41, 2) : RETURN
1130 NEXTI
1140 PRINT "CAN'T DO THAT" : S=1 : RETURN
1150 N$(6)="OPEN JAIL CELL" : RETURN
```

SINTEZA 20

Programe pentru exemplele din conversaţiile 1–14

În această sinteză sînt reunite programele aparţinînd exemplelor comentate în cadrul conversaţiilor din volumul 1 (conversaţiile 1–14).*

Sinteza 20 poate fi lecturată atît în conjuncţie cu volumele 1 şi 2, funcţie de trimiterile din conversaţii prin simbolul "vol. 2, pag. XXX" cît şi independent.

Atragem atenţia că exemplele realizate pe sistemele de calcul INDEPENDENT, CORAL pot fi urmărite de asemenea şi de către utilizatorii calculatoarelor PDP-11. Pentru posesorii calculatoarelor personale IBM PC recomandăm consultarea mementoului prezentat în sinteza 17. De asemenea, posesorii calculatoarelor APPLE, ATARI, TI, TRS-80 pot consulta sinteza de complemente informatice (21).

La reproducerea programelor s-a avut în vedere respectarea următoarei secvenţe: **EXEMPLELE 1–6** (BASIC-aMIC, BASIC-PRAE, BASIC HC-85, TIM S, SPECTRUM, BASIC-AMSTRAD, BASIC-COMMODORE, BASIC-80, BASIC-PLUS, ABASIC), **EXEMPLELE 7** (BASIC-AMSTRAD, BASIC-COMMODORE, BASIC-80, BASIC-PLUS, ABASIC), **EXEMPLELE 8** (BASIC-AMSTRAD, BASIC-80), **EXEMPLELE 9–10** (BASIC-PLUS), **EXEMPLELE 11** (BASIC-AMSTRAD, BASIC-80, BASIC-PLUS), **EXEMPLELE 12** (BASIC-AMSTRAD, **EXEMPLUL 13** (BASIC-AMSTRAD), **TEMA 14** (BASIC HC-85, TIM S, SPECTRUM).

De notat că rezultatele execuţiei programelor apar în cadrul fiecărei conversaţii.

---

* La realizarea unora dintre programe autorul a colaborat cu: ing. Victor Stoicea, mat. Cristian Marinoiu, ing. Marius Manea.

EXEMPLELE 1

BASIC - aMIC

```
5 INIT
10 PRINT "INVATAM BASIC..."
15 PRINT
20 PRINT "...CONVERSIND CU CAL
CULATORUL "
99 END
```

BASIC - PRAE

```
5 CLS
10 PRINT " INVATAM BASIC.."
15 PRINT
20 PRINT "...CONVERSIND CU CA
LCULATORUL "
```

BASIC HC-85,TIM S,SPECTRUM

```
5 CLS
10 PRINT "Invatam BASIC..."
15 PRINT
20 PRINT "...conversind cu cal
culatorul"
```

BASIC-AMSTRAD

```
3 mode 2
5 cls
10 print "Invatam BASIC..."
15 print
20 print "...conversind cu calculatorul"
```

BASIC-COMMODORE

```
5 PRINT CHR$(I47)
10 PRINT "INVATAM BASIC..."
15 PRINT
20 PRINT "...CONVERSIND CU CALCULATORUL"
```

BASIC-80

```
5  PRINT CHR$(24)
10 PRINT "Invatam BASIC..."
15 PRINT
20 PRINT "...conversind cu calculatorul"
```

BASIC-PLUS

```
10 PRINT "INVATAM BASIC..."
15 PRINT
20 PRINT "...CONVERSIND CU CALCULATORUL"
99 END
```

ABASIC

```
5  CLEAR$
10 PRINT "INVATAM BASIC..."
15 PRINT
20 PRINT "...CONVERSIND CU CALCULATORUL"
90 END
```

EXEMPLELE 2

BASIC-aMIC

```
SCR
10 R=3
20 A=4*PI*R^2
30 PRINT A
40 PRINT "A=",A
50 PRINT "A=";A
99 END
```

BASIC-PRAE

```
NEW
10 R=3
20 A=4*PI*R^2
30 PRINT A
40 PRINT "A=",A
50 PRINT "A=";A
60 PRINT 4;"*";PI;"*";R;"^";2
;"=";A
70 PRINT 4;"*";PI;"*";R;"^";2
;"=";4*PI*R^2
80 END
```

BASIC ПС-85,TIM S,SPECTRUM

```
NEW
  10 LET R=3
  20 LET A=4*PI*R^2
  30 PRINT A
  40 PRINT "A=",A
  50 PRINT "A=";A
  60 PRINT "A"'A
  70 PRINT "4*3.14*";R;"^";2;"="
;A
  80 PRINT 4;"*";PI;"*";R;"^";2;
"=";4*PI*R^2
9999 STOP
```

BASIC – AMSTRAD

```
NEW
5  MODE 2
10 r=3
20 a=4*PI*r^2
25 PRINT a
30 PRINT "A=",a
35 PRINT 4;"*";PI;"*";r;"^";"2";"=";a
40 PRINT 4;"*";PI;"*";r;"^";"2";"=";4*PI*r^2
```

BASIC – COMMODORE

```
NEW
10 R=3
20 A=4*π*R^2
30 ?A
40 ?"A=",A
50 ?"A=";A
60 ?4"x"π"x"R"^"2"="A
70 ?4"x"π"x"R"^"2"="4*π*R^2
```

BASIC – 80

```
NEW
10 PI=3.14159
20 R=3
30 A=4*PI*R^2
40 PRINT A
50 PRINT "A=",A
60 PRINT "A=";A
70 PRINT 4"*"PI"*"R"^"2"="A
80 PRINT 4"*"PI"*"R"^"2"="4*PI*R^2
```

BASIC – PLUS

```
NEW
10 R=3
20 A=4*PI*R**2
30 PRINT A
40 PRINT "A=",A
50 PRINT "A=";A
60 PRINT 4;"*";PI;"*";R;"^";2;"=";A
70 PRINT 4;"*";PI;"*";R;"^";2;"=";4*PI*R^2
1000 END
```

ABASIC

```
NEW
10   R=3
20   A=4*PI*R**2
30   PRINT A
40   PRINT 'A=',A
50   PRINT 'A=';A
60   PRINT 4;'*';PI;'*';R;'^';2;'=';A
70   PRINT 4;'*';PI;'*';R;'^';2;'=';4*PI*R^2
1000 END
```

EXEMPLELE 3

BASIC-aMIC

```
SCR
10  REM "EXEMPLUL 3-PC(AMIC)"
20  REM "PROGRAMUL CITESTE RAZA
(R)"
30  REM "A UNUI REZERVOR SFERIC
,"
40  REM "CALCULEAZA ARIA (A),"
50  REM "VOLUMUL (V) SI TIPAREST
E:"
60  REM "RAZA,ARIA,VOLUMUL"
70  PRINT "RAZA=";
80  INPUT R
90  A=4*PI*R^2
100 V=4*PI*R^3/3
110 PRINT "R=";R;" A=";A;" V="
;V
120 END
```

BASIC - PRAE

```
NEW
10 REM PROGRAMUL CITESTE RAZA
(R) A UNUI REZERVOR SFERIC,
CALCULEAZA ARIA(A),VOLUMUL(V)
SI TIPARESTE: RAZA,ARIA,VOLUM
UL
20 PRINT "RAZA=";
```

```
30 INPUT R
40 A=4*PI*R^2
50 V=4*PI*R^3/3
60 PRINT "R=";R;" A=";A;" V="
;V
70 END
```

BASIC HC-85,TIM S,SPECTRUM

```
NEW
  10 REM PROGRAMUL CITESTE RAZA
(R) A UNUI REZERVOR SFERIC, CALC
ULEAZA ARIA (A),VOLUMUL (V) SI T
IPARESTE : RAZA,ARIA,VOLUMUL
  20 PRINT "RAZA=";
  30 INPUT R
  40 LET A=4*PI*R^2
  50 LET V=4*PI*R^3/3
  60 PRINT "R=";R;" A=";A;" V=";
V
  70 STOP
```

BASIC – AMSTRAD

```
NEW
5   CLS
10  REM programul citeste raza (R) a unui
rezervor sferic,calculeaza aria (A),volumul
(V) si tipareste: raza,aria,volumul
20  PRINT "RAZA = ";
30  INPUT r
40  r%=r
50  PRINT
55  ZONE 20
60  PRINT "RAZA (Nr.real)","RAZA (Nr.intreg)"
70  PRINT r,r%
80  a = 4*PI*r^2
90  v = 4*PI*r^3/3
100 PRINT
110 PRINT "R=";r;" A=";a;" V=";v
120 PRINT
130 STOP
140 END
```

BASIC COMMODORE

```
NEW
5   REM PROGRAMUL CITESTE RAZA(R) A UNUI
REZERVOR SFERIC,CALCULEAZA ARIA (A),VOLU
MUL (V) SI TIPARESTE :RAZA,ARIA,VOLUMUL
10  PRINT CHR$(147)
20  PRINT ;RAZA=" ;
30  INPUT r
40  r%=r
50  PRINT
60  PRINT "RAZA(Nr.real)";TAB(21);"RAZA (N
r.intreg)"
65  PRINT
70  PRINT r;TAB(21);r%
80  a=4*π*r^2
90  v=4*π*r^3/3
100 PRINT
110 PRINT "R=";r;" A=";a;" V=";v
120 PRINT
130 STOP
140 END
```

BASIC-80

```
NEW
5   PRINT CHR$(24)
10  REM Programul citeste raza (R) a unui
15  REM rezervor sferic,
20  REM calculeaza aria (A),volumul (V) si
30  REM tipareste:Raza,Aria,Volumul
35  PI=3.14159
40  PRINT "Raza=";
45  INPUT R
50  R%=R
60  PRINT "RAZA(Nr. real)";SPC(7);"RAZA(Nr.intreg)"
65  PRINT
70  PRINT R;SPC(15);R%
80  A=4*PI*R^2
90  V=4*PI*R^3/3
100 PRINT
110 PRINT "R=";R;" A=";A;" V=";V
120 PRINT
130 STOP
140 END
```

BASIC - PLUS

```
100 REM PROGRAMUL CITESTE RAZA (R) A UNUI
    REZERVOR SFERIC,
110 REM CALCULEAZA ARIA (A) , VOLUMUL (V)
    SI TIPARESTE :
120 REM RAZA , ARIA , VOLUMUL
140 PRINT "RAZA=";
150 INPUT R
160 LET R%=R
170 PRINT
180 PRINT "RAZA (NR.REAL)";TAB(40);"RAZA (NR.INTREG)"
190 PRINT R;TAB(40);R%
200 LET A = 4*PI*R**2
210 LET V = 4*PI*R**3/3
220 PRINT
230 PRINT "R=";R,"A =" ;A,"V = ";V
240 PRINT
250 STOP
260 END
```

ABASIC

```
NEW
10  REM PROGRAMUL CITESTE RAZA (R) A UNUI
20  REM REZERVOR SFERIC,CALCULEAZA ARIA(A),
30  REM VOLUMUL (V) SI TIPARESTE:RAZA,ARIA,
40  REM VOLUMUL
50  PRINT "RAZA=";
60  INPUT R
70  R%=R
80  PRINT
90  PRINT "RAZA(NR.REAL)";TAB(45);"RAZA(NR.INTREG)"
100 PRINT R;TAB(45);R%
110 A=4*PI*R^2
120 V=4*PI*R^3/3
130 PRINT
140 PRINT "R=";R,"A=";A,"V=";V
150 PRINT
160 PRINT
170 STOP
180 END
```

EXEMPLELE 4

BASIC-aMIC

```
SCR
100 REM "PROGRAMUL CALCULEAZA
SI AFISEAZA CANTITATEA DE BENZ
INA (S) DIN MAI MULTE REZERVOAR
E CILINDRICE ECHILATERALE (GENE
RATOAREA ESTE EGALA CU DIAMETR
UL) A CAROR RAZA (R) VARIAZA DE
LA 2 LA 10 METRI CU PASUL 1"
110 REM
220 REM "CITIRE (DINAMICA) DENS
ITATE BENZINA (D)"
240 PRINT "DENSITATEA BENZINEI
=";
250 INPUT D
260 REM "TIPARIRE CAP TABEL"
270 PRINT "===================
========"
280 PRINT "RAZA","  MASA"
290 PRINT "===================
========"
300 REM "CALCULUL MASEI DE BEN
ZINA DIN REZERVOARE"
330 S=0
340 FOR R=2 TO 10
350   V=2*PI*R^3
360   M=D*V
370   S=S+M
380   PRINT R,M
390 NEXT R
400 REM "TIPARESTE  MASA TOTALA
DE BENZINA"
420 PRINT "===================
========"
430 PRINT "MASA TOTALA=";S;"TO
NE"
440 END
```

BASIC - PRAE

```
NEW
100 REM PROGRAMUL CALCULEAZA
SI AFISEAZA CANTITATEA DE BEN
ZINA(S) DIN MAI MULTE REZERVO
ARE CILINDRICE ECHILATERALE
(GENERATOAREA ESTE EGALA CU D
IAMETRUL) A CAROR RAZA (R) V
ARIAZA DE LA 2 LA 10 METRI CU
 PASUL 1
110 REM
220 REM CITIRE (DINAMICA) DEN
SITATE BENZINA (D)
240 INPUT "D=";D
250 PRINT
260 REM TIPARIRE CAP TABEL
270 PRINT "==================
========"
280 PRINT "RAZA","MASA"
290 PRINT "==================
========"
300 REM CALCULUL MASEI DE BEN
ZINA DIN REZERVOARE
330 S=0
340 FOR R=2 TO 10
350   V=2*PI*R^3
360   M=D*V
370   S=S+M
380   PRINT R,M
390 NEXT R
400 REM TIPARESTE MASA TOTALA
DE BENZINA
420 PRINT "==================
========"
430 PRINT "MASA TOTALA=";S;"T
ONE"
440 END
```

BASIC HC - 85,TIMS,SPECTRUM

```
  10 REM PROGRAMUL CALĈULEAZA S
I AFISEAZA CANTITATEA DE BENZINA
(S) DIN MAI MULTE REZERVOARE
CILINDRICE ECHILATERALE (GENERAT
OAREA ESTE EGALA CU DIAMETRUL) A
CAROR RAZA (R) VARIAZA DE LA 2 L
A 10 m CU PASUL 1
  20 REM CITIRE (DINAMICA) DENS
ITATE BENZINA (D)
```

```
  30  INPUT "DENSITATEA BENZINEI
="; D
  40  REM TIPARESTE CAP TABEL
  50  PRINT "RAZA","MASA"
  60  REM TIPARESTE RIND DE SPAT
II
  70  PRINT
  80  REM CALCULUL MASEI DE BENZ
INA DIN REZERVOARE
  90  LET S=0
 100  FOR R=2 TO 10
 110    LET V=2*PI*R^3
 120    LET M=D*V
 130    LET S=S+M
 140    PRINT R,M
 150  NEXT R
 160  REM TIPARESTE MASA TOTALA
DE BENZINA
 170  PRINT "--------------------
--------"
 180  PRINT "   ";"MASA TOTALA="
,S;"TONE"
 190  STOP
```

BASIC – AMSTRAD

```
NEW
5   REM PROGRAMUL CALCULEAZA SI AFISEAZA
CANTITATEA TOTALA DE BENZINA (s) DIN MAI MULTE
REZERVOARE CILINDRICE ECHILATERALE (GENE
RATOAREA ESTE EGALA CU DIAMETRUL) A CARO
R RAZA (r) VARIAZA DE LA 2 LA 10 METRI
CU PASUL 1
10  INPUT "DENSITATEA BENZINEI =";d
15  PRINT
16  PRINT
20  PRINT "RAZA";SPC(15);"MASA"
25  FOR i=1 TO 40
26    PRINT "-";
27  NEXT
30    s=0
40  FOR r=2 TO 10
50    V=2*PI*r^3
60    m=d*v
70    s=s+m
80    PRINT USING "##";r;:PRINT SPC(12);:
PRINT USING "#######.#######";m
```

```
85  NEXT r
90  FOR i=1 TO 40
92    PRINT "-";
94  NEXT i
100 PRINT "MASA TOTALA=";:PRINT SPC(3) ;:
PRINT USING "######.#######";s;:PRINT "T
ONE"
110 END
```

BASIC - COMMODORE

```
NEW
5   REM PROGRAMUL CALCULEAZA SI AFISEAZA
CANTITATEA DE BENZINA (s) DIN MAI MULTE
REZERVOARE CILINDRICE ECHILATERALE(GENER
ATOAREA ESTE EGALA CU DIAMETRUL) A CAROR
RAZA (r) VARIAZA DE LA 2 LA 10 METRI CU
PASUL 1
10  INPUT "DENSITATEA BENZINEI=";d
20  PRINT "raza","masa"
25  PRINT"------------------------------
----------"
30  s=0
40  FOR r=2 TO 10
50    v=2*π*r^3
60    m=d*v
70    s=s+m
80    PRINT r,m
85  NEXT r
90  PRINT "------------------------------
----------"
100 PRINT " ";"masa totala=",s;"tone"
```

BASIC - 80

```
10 REM "Programul calculeaza si afiseaza
   cantitatea de benzina(s)din mai multe"
20 REM "rezervoare cilindrice echilaterale
   (generatoarea este egala cu"
30 REM "diametrul) a caror raza (R) variaza
   de la 2 la 10 metri cu pasul 1"
40 REM
50 REM "Citire (dinamica)densitate benzina (D)"
60 INPUT "-Densitatea benzinei=";D
70 REM "Tipareste cap tabel"
75 PRINT "========================"
80 PRINT "Raza","Masa"
85 PRINT "========================"
```

```
110 REM "Calculul masei de benzina din rezervoare"
115 PI=4*ATN(1):REM"Numarul PI"
120 S=0
130 FOR R=2 TO 10
140   V=2*PI*R^3
150   M=D*V
160   S=S+M
170   PRINT R,M
180 NEXT R
190 REM "Tipareste masa totala de benzina"
200 PRINT "======================"
210 PRINT "Masa totala=";S;"tone"
220 END
```

BASIC - PLUS

```
100 REM PROGRAMUL CALCULEAZA SI AFISEAZA
    CANTITATEA DE BENZINA(S)
110 REM DIN MAI MULTE REZERVOARE CILINDRICE
    ECHILATERALE (GENERATOAREA
120 REM ESTE EGALA CU DIAMETRUL),A CAROR RAZA
    (R) VARIAZA DE LA
130 REM 2 LA 10 METRI CU PASUL 1
140 REM
150 REM CITIRE (DINAMICA) DENSITATE BENZINA (D)
160 INPUT "-DENSITATEA BENZINEI=";D
170 REM TIPARESTE CAP TABEL
180 PRINT "============================="
190 PRINT "RAZA","MASA"
200 FOR I=1 TO 29
202   PRINT TAB (I);"=";
204 NEXT I
210 PRINT
220 REM CALCULUL MASEI DE BENZINA DIN REZERVOARE
230 S=0 ! MASA TOTALA
240 FOR R=2 TO 10 ! INITIALIZARE CICLU
250   V=2*PI*R**3 ! CALCUL VOLUM
260   M=D*V ! CALCUL MASA
270   S=S+M ! ADUNARE LA MASA TOTALA
280   PRINT USING "##      ######.##",R;M
290 NEXT R ! INCHIDERE CICLU
300 A$= "="! INIT. VARIABILA ALFA
310 PRINT A$;; FOR I=1 TO 29
320 PRINT
330 PRINT
340 PRINT "T O T A L=";S;"TONE"
350 PRINT
```

```
360 PRINT "=";; FOR I=1 TO 29
370 PRINT FOR I=1 TO 3 ! SARE 3 RINDURI
380 STOP
390 END
```

ABASIC

```
100 REM PROGRAMUL CALCULEAZA SI ÀFISEAZA
    CANTITATEA DE BENZINA
110 REM DIN MAI MULTE REZERVOARE CILINDRICE
    ECHILATERALE
120 REM (GENERATOAREA ESTE EGALA CU DIAMETRUL),
    A CAROR RAZA (R)
130 REM VARIAZA DE LA 2 LA 10 METRI CU PASUL 1
140 REM
150 REM CITIRE (DINAMICA) DENSITATE BENZINA (D)
160 INPUT "DENSITATEA BENZINEI=";D
170 REM TIPARESTE CAP TABEL
180 PRINT "========================="
190 PRINT "RAZA","MASA"
200 FOR I=1 TO 29
202   PRINT TAB (I);"=";
204 NEXT I
210 PRINT
220 REM CALCULUL MASEI DE BENZINA DIN REZERVOARE
230 S=0
240 FOR R=2 TO 10
250   V=2*PI*R**3
260   M=D*V
270   S=S+M
280   PRINT USING "##            ######.##",R;M
290 NEXT R
300 A$="="
310 FOR I=1 TO 29
315   PRINT A$;
318 NEXT I
320 PRINT
330 PRINT
340 PRINT "TOTAL=";S;"TONE"
350 PRINT
360 FOR I=1 TO 29
365   PRINT "=";
368 NEXT I
370 FOR I=1 TO 3
375   PRINT
378 NEXT I
380 STOP
390 END
```

EXEMPLELE 5

BASIC-aMIC

```
10 REM PROGRAMUL CALCULEAZA SI
AFISEAZA CANTITATEA DE BENZINA
(S) DINTR-UN NUMAR OARECARE (N
) DE REZERVOARE CILINDRICE ECH
ILATERALE (ACEST NUMAR ESTE FU
RNIZAT CA PARAMETRU).PROGRAMUL
CITESTE  RAZELE REZERVOARELOR
INTR-UN VECTOR (R) PRINTR-O PR
OCEDURA DE INTRODUCERE DINAMIC
A A DATELOR ( RAZELE NEGATIVE
NU SE IAU IN CONSIDERARE). PRO
GRAMUL CITESTE DENSITATEA BENZ
INEI (D) PRINTR-O PROCEDURA DE
INTRODUCERE STATICA A DATELOR.
20  PRINT "NUMAR  REZERVOARE";
30  INPUT N
40  PRINT N
50  REM CITIRE (DINAMICA) SI V
ALIDARE DATE
60  DIM R(30)
70  FOR I=1 TO N
80    PRINT "R(";I;")=";
90    INPUT R(I)
100   PRINT R(I)
110   IF R(I)>0 THEN 170
120   PRINT "R(";I;")=";R(I);"
RAZA NEGATIVA"
130   PRINT "R(";I;")=";
140   INPUT R(I)
150   PRINT R(I)
160   GO TO 110
170 NEXT I
180 REM CITIRE (STATICA) DENSI
TATE BENZINA
190 READ D
200 REM  CALCULUL MASEI DE BEN
ZINA DIN REZERVOARE
210 PRINT
220 REM TIPARIRE CAP  TABEL
230 PRINT "RAZA"," MASA"
240 S=0
250 FOR I=1 TO N
260   M=D*2*PI*R(I)^3
270   S=S+M
280   PRINT R(I),M
290 NEXT I
300 FOR I=1 TO 25
310   PRINT "-";
320 NEXT I
330 REM TIPARIRE REZULTAT FINA
L
340 PRINT "MASA TOTALA=",S;"TO
NE"
350 DATA .7
360 END
```

BASIC-PRAE

```
10 REM PROGRAMUL CALCULEAZA S
I AFISEAZA CANTITATEA DE BENZ
INA (S) DINTR-UN NUMAR OARECA
RE (N) DE REZERVOARE CILINDRI
CE ECHILATERALE (ACEST NUMAR
ESTE FURNIZAT CA PARAMETRU).
PROGRAMUL CITESTE RAZELE REZE
RVOARELOR INTR-UN VECTOR (R)
PRINTR-O PROCEDURA DE INTRODU
CERE DINAMICA A DATELOR (RAZE
LE NEGATIVE NU SE IAU IN CONS
IDERARE).PROGRAMUL CITESTE DE
NSITATEA BENZINEI (D) PRINTR-
O PROCEDURA DE INTRODUCERE ST
ATICA A DATELOR.
20 INPUT "NUMAR REZERVOARE";N
40 PRINT N
50 REM CITIRE (DINAMICA) SI V
ALIDARE DATE
60 DIM R(N)
70 FOR I=1 TO N
80    PRINT "R(";I;")=";
90    INPUT R(I)
100   PRINT R(I)
110   IF R(I)>0 THEN 170
120   PRINT "R(";I;")=";R(I);
"RAZA NEGATIVA"
130   PRINT "R(";I;")=";
140   INPUT R(I)
```

```
150   PRINT R(I)
160   GO TO 110
170 NEXT I
180 REM CITIRE (STATICA) DENS
ITATE BENZINA "
190 READ D
200 REM CALCULUL MASEI DE BEN
ZINA DIN REZERVOARE
210 PRINT
220 REM TIPARIRE CAP  TABEL
230 PRINT "RAZA"," MASA"
232 FOR I=1 TO 28
234   PRINT "=";
236 NEXT I
240 S=0
250 FOR I=1 TO N
260   M=D*2*PI*R(I)^3
270   S=S+M
280   PRINT R(I),M
290 NEXT I
300 FOR I=1 TO 28
310   PRINT "-";
320 NEXT I
330 REM TIPARIRE REZULTAT FIN
AL
340 PRINT "MASA TOTALA=",S;"T
ONE"
350 DATA .7
360 END
```

BASIC HC-85,TIMS,SPECTRUM

```
 10 REM Programul calculeaza si
afiseaza cantitatea de benzina (
S) dintr-un numar oarecare (N) d
e rezervoare cilindrice echilate
rale (acest numar este furnizat
ca parametru).Programul citeste
razele rezervoarelor intr-un vec
tor (R) printr-o procedura de in
troducere dinamica a datelor (ra
zele negative nu se iau in consi
derare). Progranul citeste densi
tatea benzinei(D) printr-o proce
dura de introducere statica a da
telor.
 20 PRINT "Numar rezervoare =";
 30 INPUT N
 40 PRINT N
 50 REM Citire (dinamica) si va
lidare date
 60 DIM R(N)
 70 FOR I=1 TO N
 80    PRINT "R(";I;")=";
 90    INPUT R(I)
100    PRINT R(I)
110    IF R(I)>0 THEN GO TO 140
120    PRINT "R(";I;")=";R(I);"R
aza negativa"
130    GO TO 80
140 NEXT I
150 REM Citire (statica) densit
ate benzina
160 READ D
170 REM  Calculul masei de benz
ina din rezervoare
180 PRINT '"Calculul masei tota
le de benzina din ";N;" rezervoa
re cilindrice echilaterale"
190 REM Tiparire cap de tabel
200 PRINT '"Raza"," Masa"''
210 LET S=0
220 FOR I=1 TO N
230    LET M=D*2*PI*R(I)^3
240    LET S=S+M
250    PRINT R(I),M
260 NEXT I
270 FOR N=1 TO 25
280    PRINT "-";
290 NEXT N
300 REM Tiparire rezultat final
310 PRINT ,"Masa totala =";S;"
tone"
320 DATA .7
330 STOP
```

BASIC-AMSTRAD

```
5     MODE 2
10    REM "Programul calculeaza si afiseaza
      cantitatea de benzina (S) dintr-un"
20    REM "numar oarecare (N) de rezervoare
      cilindrice echilaterale(acest numar"
30    REM "este furnizat ca parametru ).
      Programul citeste razele rezervoarelor"
40    REM "intr-un vector (R) printr-o procedura
      de introducere dinamica a"
50    REM "datelor (razele negative nu se iau in
      considerare). Progranul citeste"
60    REM "densitatea benzinei(D) printr-o procedura
      de introducere statica"
70    REM "a datelor. "
80    PRINT "Numar de rezervoare =";
90    INPUT N
100   PRINT N
110   REM "Citire (dinamica) si validare date"
120   DIM R(N)
130   FOR I=1 TO N
140      PRINT "R(";I;")=";
150      INPUT R(I)
160      PRINT R(I)
170      WHILE R(I)<0
180         PRINT "R(";I;")=";R(I);
190         PRINT "R(";I;")=";
200         INPUT R(I)
210         PRINT R(I)
220      WEND
230   NEXT I
240   REM "Citire (statica) densitate benzina "
245   DATA .7
250   READ D
260   REM "Calculul masei de benzina din rezervoare"
270   PRINT
280   REM  "Calculul masei totale de benzina"
290   REM "Tiparire cap de tabel"
300   PRINT "Raza","Masa"
310   S=0
320   FOR I=1 TO N
330      M=D*2*PI*R(I)^3
340      S=S+M
350      PRINT R(I),M
360   NEXT I
370   FOR I=1 TO 25
```

```
380       PRINT "-";
390     NEXT I
395     PRINT
400     REM "Tipareste rezultat final"
410     PRINT "Masa totala =",S;" tone"
420     DATA .7
430     END
```

BASIC-COMMODORE

```
10 REM Programul calculeaza si afiseaza
cantitatea de benzina (S) dintr-un numar
oarecare (N) de rezervoare cilindrice ec
hilaterale (acest numar este furnizat ca
parametru).Programul citeste razele reze
rvoarelor intr-un vector (R) printr-o pr
ocedura de introducere dinamica a datelo
r (razele negative nu se iau in consider
are). Programul citeste densitatea benzi
nei(D) printr-o procedura de introducere
statica a datelor.
20 PRINT "Numar rezervoare";
30 INPUT N
40 PRINT N
50 REM Citire(dinamica)si validare date
60 DIM R(N)
70 FOR I=1 TO N
80   PRINT "R(";I;")=";
90   INPUT R(I)
100  PRINT R(I)
110  IF R(I)>0 THEN 170
120  PRINT "R(";I;")=";R(I);"Raza negati
va"
130  PRINT "R(";I;")=";
140  INPUT R(I)
150  PRINT R(I)
160  GO TO 110
170 NEXT I
180 REM Citire (statica) densitate benzi
na
190 READ D
200 REM  Calculul masei de benzina din r
ezervoare
210 PRINT
220 REM Tiparire cap  tabel
230 PRINT "Raza"," Masa"
```

```
240 S=0
250 FOR I=1 TO N
260   M=D*2*π*R(I)^3
270   S=S+M
280   PRINT R(I),M
290 NEXT I
300 FOR I=1 TO 25
310   PRINT "-";
320 NEXT I
330 REM Tiparire rezultat final
340 PRINT "Masa totala=",S;"tone"
350 DATA .7
360 END
```

BASIC-80

```
10    REM "Programul calculeaza si afiseaza
      cantitatea de benzina (S) dintr-un"
20    REM "numar oarecare (N) de rezervoare
      cilindrice echilaterale(acest numar"
30    REM "este furnizat ca parametru).
      Programul citeste razele rezervoarelor"
40    REM "intr-un vector (R) printr-o procedura
      de introducere dinamica a"
50    REM "datelor (razele negative nu se iau in
      considerare). Progranul citeste"
60    REM "densitatea benzinei(D) printr-o procedura
      de introducere statica"
70    REM "a datelor. "
80    PRINT "Numar de rezervoare =";
90    INPUT N
100   PRINT N
110   REM "Citire (dinamica) si validare date"
120   DIM R(N)
130   FOR I=1 TO N
140     PRINT "R(";I;")=";
150     INPUT R(I)
160     PRINT R(I)
170     WHILE R(I)<0
180       PRINT "R(";I;")=";R(I);" Raza negativa"
190       PRINT "R(";I;")=";
200       INPUT R(I)
210       PRINT R(I)
220     WEND
230   NEXT I
```

```
240  REM "Citire (statica) densitate benzina "
245  DATA .7
250  READ D
260  REM "Calculul masei de benzina din rezervoare"
270  PRINT "Calculul masei totale de benzina"
280  PRINT "din ";N;" rezervoare cilindrice echilaterale"
290  REM "Tipareste cap de tabel"
300  PRINT "Raza"," Masa"
310  S=0
315  PI=4*ATN(1)
320  FOR I=1 TO N
330    M=D*2*PI*R(I)^3
340    S=S+M
350    PRINT R(I),M
360  NEXT I
370  FOR N=1 TO 25
380     PRINT "-";
390  NEXT N
395  PRINT
400  REM "Tipareste rezultat final"
410  PRINT "Masa totala =";S;" tone"
420  DATA .7
430  END
```

BASIC-PLUS

```
10   REM Programul calculeaza si afiseaza cantitatea
     de benzina (S) dintr-un
20   REM numar oarecare (N) de rezervoare cilindrice
     echilaterale(acest numar
30   REM este furnizat ca parametru ).
     Programul citeste razele rezervoarelor
40   REM intr-un vector (R) printr-o procedura de
     introducere dinamica a
50   REM datelor (razele negative nu se iau in
     considerare). Progranul citeste
60   REM densitatea benzinei(D) printr-o procedura de
     introducere statica
70   REM a datelor.
80   INPUT "Numar de rezervoare =";N
85   PRINT
90   PRINT "Ati specificat  ";N;"rezervoare"
100  REM Citire dinamica si validare date
110  DIM R(N)
```

```
120   FOR I=1 TO N
130     PRINT "Rezervorul numarul";I
140     INPUT "Raza rezervorului ";R(I)
150     PRINT "Ati specificat ";R(I)
160     PRINT
170     IF R(I)>0 THEN 200
180     PRINT "Eroare de introducere date - Raza negativa -
        Reluare"
190     GO TO 130
200   NEXT I
210   REM Citire (statica) densitate benzina
220   READ D
230   REM Calculul masei de benzina din rezervoare
240   PRINT "Calculul masei totale de benzina"
250   PRINT "din ";N;" rezervoare cilindrice echilaterale"
260   REM Tipareste cap de tabel
270   PRINT
280   PRINT "RAZA"," MASA"
290   PRINT
300   S=0
310   FOR I=1 TO N
320     M=D*2*PI*R(I)**3
330     S=S+M
340     PRINT R(I),M
350   NEXT I
360   FOR N=1 TO 25
370     PRINT "-";
380   NEXT N
390   PRINT
400   REM Tipareste rezultat final
410   PRINT "MASA TOTALA =";S;" TONE"
420   DATA .7
430   PRINT
440   STOP
450   END
```

ABASIC

```
10    REM "Programul calculeaza si afiseaza
      cantitatea de benzina (S) dintr-un"
20    REM "numar oarecare (N) de rezervoare
      cilindrice echilaterale(acest numar"
30    REM "este furnizat ca parametru ).
      Programul citeste razele rezervoarelor"
40    REM "intr-un vector (R) printr-o procedura
      de introducere dinamica a"
```

```
50    REM "datelor (razele negative nu se iau in
      considerare). Progranul citeste"
60    REM "densitatea benzinei(D) printr-o
      procedura de introducere statica"
70    REM "a datelor. "
80    PRINT "Numar de rezervoare =";
90    INPUT N
100   PRINT N
110   REM "Citire (dinamica) si validare date"
120   DIM R(N)
130   FOR I=1 TO N
140     PRINT "R(";I;")=";
150     INPUT R(I)
160     PRINT R(I)
170     WHILE R(I)<0
180       PRINT "R(";I;")=";R(I);" Raza negativa"
190       PRINT "R(";I;")=";
200       INPUT R(I)
210       PRINT R(I)
220     WEND
230   NEXT I
240   REM "Citire (statica) densitate benzina "
245   DATA .7
250   READ D
260   REM "Calculul masei de benzina din rezervoare"
270   PRINT "Calculul masei totale de benzina"
280   PRINT "din ";N;" rezervoare cilindrice echilaterale"
290   REM "Tipareste cap de tabel"
300   PRINT "Raza"," Masa"
310   S=0
315   PI=4*ATN(1)
320   FOR I=1 TO N
330     M=D*2*PI*R(I)^3
340     S=S+M
350     PRINT R(I),M
360   NEXT I
370   FOR N=1 TO 25
380     PRINT "-";
390   NEXT N
395   PRINT
400   REM "Tipareste rezultat final"
410   PRINT "Masa totala =";S;" tone"
420   DATA .7
430   END
```

EXEMPLELE 6

BASIC-aMIC

```
5   DIM C$(6,8),B(6),D(3),A(6
,3)
9   FOR I=1 TO 6
10   READ C$(I)
12  NEXT I
15  DATA LUNI,MARTI,MIERCURI,
JOI,VINERI,SIMBATA
80  INIT
85  PRINT AT (8,4);"INTRODUCE
TI LIVRARILE"
90  PRINT AT (10,4);"DE BENZI
NA PE  ZILE"
95  PRINT AT (12,4);"IN ORDIN
EA  R1  R2  R3  "
100 MAT INPUT A(6,3)
200 FOR Z=1 TO 6
205   S=0
210   FOR I=1 TO 3
215     S=S+A(Z,I)
220   NEXT I
225   B(Z)=S/3
230 NEXT Z
300 FOR I=1 TO 3
305   S=0
310   FOR Z=1 TO 6
315     S=S+A(Z,I)
320   NEXT Z
325   D(I)=S/6
330 NEXT I
400 INIT
405 PRINT AT (7,3);"ZIUA    R
1  R2  R3  MEDIA"
410 FOR L=1 TO 30
415   PRINT AT (8,L);"-"
420 NEXT L
500 FOR Z=1 TO 6
505   P=8+Z
510   PRINT AT (P,1);C$(Z)
515   FOR I=1 TO 3
520     Q=5*I
525     PRINT AT (P,Q);A(Z,I)
530   NEXT I
535   PRINT AT (P,26);B(Z)
540 NEXT Z
600 FOR M=1 TO 30
605   PRINT AT (16,M);"-"
610 NEXT M
700 PRINT AT (17,3);"MEDIA "
705 FOR I=1 TO 3
710   Q=5*(I+1)
715   PRINT AT (17,Q),D(I)
720 NEXT I
800 H=A(1,1)
805 R=1
810 C=1
815 FOR Z=1 TO 6
820   FOR I=1 TO 3
825     IF A(Z,I)<=H THEN 845
830     H=A(Z,I)
835     R=Z
840     C=I
845   NEXT I
850 NEXT Z
855 PRINT AT (20,1);"MAXIMUL
LIVRARILOR = ";H;" TONE "
860 PRINT AT (22,4);"REZERVOR
 = R";C
865 PRINT AT (24,4);"ZIUA = "
;C$(R)
900 END
```

BASIC-PRAE

```
5   DIM C$(6,8),B(6),D(3),A(6
,3)
9   FOR I=1 TO 6
10   READ C$(I)
12  NEXT I
15  DATA LUNI,MARTI,MIERCURI,
JOI,VINERI,SIMBATA
20  CLS
100 FOR Z=1 TO 6
110   PRINT C$(Z);" :"
120   FOR I=1 TO 3
```

```
130       PRINT "R";
135       PRINT USING "#";I;
138       PRINT " ";
140       INPUT A(Z,I)
150    NEXT I
160 NEXT Z
200 FOR Z=1 TO 6
205    S=0
210    FOR I=1 TO 3
215    S=S+A(Z,I)
220    NEXT I
225    B(Z)=S/3
230 NEXT Z
300 FOR I=1 TO 3
305    S=0
310    FOR Z=1 TO 6
315      S=S+A(Z,I)
320    NEXT Z
325    D(I)=S/6
330 NEXT I
400 CLS
405 PRINT "ZIUA     R1   R2
R3   MEDIA"
410 FOR L=1 TO 28
415    PRINT "-";
420 NEXT L
500 FOR Z=1 TO 6
505    PRINT C$(Z);
510    FOR I=1 TO 3
515      PRINT TAB(5*I+5);A(Z,
I);
520    NEXT I
535    PRINT TAB(23);B(Z);
540    PRINT
545 NEXT Z
600 FOR M=1 TO 28
610    PRINT "-";
620 NEXT M
630 PRINT
700 PRINT "MEDIA ";
705 FOR I=1 TO 3
710    PRINT TAB(5*I+5);D(I);
720 NEXT I
725 PRINT
800 H=A(1,1)
805 R=1
810 C=1
815 FOR Z=1 TO 6
820    FOR I=1 TO 3
825      IF A(Z,I)<=H THEN 845
830      H=A(Z,I)
835      R=Z
840      C=I
845    NEXT I
850 NEXT Z
855 PRINT
860 PRINT "MAXIMUL LIVRARILOR
= ";H;" TONE "
865 PRINT "REZERVOR = R";C
870 PRINT "ZIUA = ";C$(R)
900 END
```

BASIC HC-85,TIMS,SPECTRUM

```
  5 DIM C$(6,8)
  6 DIM B(6)
  7 DIM D(3)
  8 DIM A(6,3)
  9 FOR I=1 TO 6
 10    READ C$(I)
 12 NEXT I
 15 DATA "LUNI","MARTI","MIERCU
RI","JOI","VINERI","SIMBATA"
 20 CLS
100 FOR Z=1 TO 6
110    PRINT C$(Z);" :"
115    FOR I=1 TO 3
120      PRINT "A(";Z;",";I;")=
";
125      INPUT A(Z,I)
130    NEXT I
140    PRINT
150 NEXT Z
200 FOR Z=1 TO 6
205    LET S=0
210    FOR I=1 TO 3
215      LET S=S+A(Z,I)
220    NEXT I
225    LET B(Z)=S/3
230 NEXT Z
300 FOR I=1 TO 3
305    LET S=0
310    FOR Z=1 TO 6
315      LET S=S+A(Z,I)
320    NEXT Z
```

```
325   LET D(I)=S/6
330 NEXT I
400 CLS
405 PRINT AT (7,3);"ZIUA    R1
 R2   R3   MEDIA"
410 FOR L=1 TO 30
415   PRINT AT (8,L);"-"
420 NEXT L
500 FOR Z=1 TO 6
505   LET P=8+Z
510   PRINT AT (P,1);C$(Z)
515   FOR I=1 TO 3
520     LET Q=5*I
525     PRINT AT (P,Q);A(Z,I)
530   NEXT I
535   PRINT AT (P,26);B(Z)
540 NEXT Z
600 FOR M=1 TO 30
605   PRINT AT (16,M);"-"
610 NEXT M
700 PRINT AT (17,3);"MEDIA "
705 FOR I=1 TO 3
710   LET Q=5*(I+1)
715   PRINT AT (17,Q),D(I)
720 NEXT I
800 LET H=A(1,1)
805 LET R=1
810 LET C=1
815 FOR Z=1 TO 6
820   FOR I=1 TO 3
825     IF A(Z,I)<=H THEN 845
830     LET H=A(Z,I)
835     LET R=Z
840     LET C=I
845   NEXT I
850 NEXT Z
855 PRINT AT (19,1);"Maximul li
vrarilor = ";H;" tone "
860 PRINT AT (20,4);"Rezervor =
R";C
865 PRINT AT (21,4);"Ziua = ";C
$(R)
900 END
```

BASIC–AMSTRAD

```
5     DIM C$(6),B(6),D(3),A(6,3)
10    FOR I=1 TO 6
15      READ C$(I)
20    NEXT I
25    DATA LUNI,MARTI,MIERCURI,JOI,VINERI,SIMBATA
30    PRINT TAB(7);"Introduceti livrarile"
40    PRINT TAB(7);"de benzina pe zile in ordinea"
45    PRINT TAB(7);"     R1    R2      R3   "
50    FOR Z=1 TO 6
51      PRINT C$(Z);" :"
55      S=0
60      FOR I=1 TO 3
61        PRINT "R";: PRINT USING "#";I;:PRINT " ";:INPUT A(Z,I)
65        S=S+A(Z,I)
70      NEXT I
75      B(Z)=S/3
80    NEXT Z
85    FOR I=1 TO 3
90      S=0
95      FOR Z=1 TO 6
100       S=S+A(Z,I)
105     NEXT Z
```

```
110   D(I)=S/6
115   NEXT I
120   PRINT "ZIUA    ","  R1","  R2","  R3","  MEDIA"
125   FOR L=1 TO 70
130     PRINT "-";
135   NEXT L
136   PRINT
140   FOR Z=1 TO 6
159     PRINT C$(Z),A(Z,1),A(Z,2),A(Z,3),B(Z)
160   NEXT Z
161   FOR M=1 TO 70
162     PRINT "-";
163   NEXT M:PRINT
164   PRINT "MEDIA         ";
165   FOR I=1 TO 3
167     PRINT D(I),
170   NEXT I
171   PRINT
175   H=A(1,1)
180   R=1
185   C=1
190   FOR Z=1 TO 6
195     FOR I=1 TO 3
200       IF A(Z,I)<=H THEN 220
205       H=A(Z,I)
210       R=Z
215       C=I
220     NEXT I
225   NEXT Z
230   PRINT "Maximul livrarilor ";H;" tone "
235   PRINT "Rezervor = R";C
240   PRINT "Ziua = ";C$(R)
245   END
```

BASIC-COMMODORE

```
5     DIM C$(6,8),B(6),D(3),A(6,3)
9     FOR I=1 TO 6
1      READ C$(I)
12    NEXT I
15    DATA LUNI,MARTI,MIERCURI,JOI,VINER
I,SIMBATA
20    CLS
100   FOR Z=1 TO 6
110     PRINT C$(Z);" :"
```

```
120      FOR I=1 TO 3
130         PRINT "A(";Z;",";I; ") =";
140         INPUT A(Z,I)
142         PRINT
150      NEXT I
160   NEXT Z
200   FOR Z=1 TO 6
205      S=0
210      FOR I=1 TO 3
215         S=S+A(Z,I)
220      NEXT I
225      B(Z)=S/3
230   NEXT Z
300   FOR I=1 TO 3
305      S=0
310      FOR Z=1 TO 6
315         S=S+A(Z,I)
320      NEXT Z
325      D(I)=S/6
330   NEXT I
400   CLS
405   PRINT "ZIUA   R1  R2   R3   MEDIA"
410   FOR L=1 TO 28
415      PRINT "-";
420   NEXT L
500   FOR Z=1 TO 6
505      PRINT C$(Z);
510      FOR I=1 TO 3
515         PRINT TAB(5*I+5);A(Z,I);
520      NEXT I
535      PRINT TAB(23);B(Z);
540      PRINT
545   NEXT Z
600   FOR M=1 TO 28
610      PRINT "-";
620   NEXT M
630   PRINT
700   PRINT "MEDIA ";
705   FOR I=1 TO 3
710      PRINT TAB(5*I+5);D(I);
720   NEXT I
725   PRINT
800   H=A(1,1)
805   R=1
810   C=1
815   FOR Z=1 TO 6
820      FOR I=1 TO 3
```

```
825          IF A(Z,I)<=H THEN 845
830          H=A(Z,I)
835          R=Z
840          C=I
845        NEXT I
850      NEXT Z
855      PRINT
860      PRINT "Maximul livrarilor = ";H;"
tone "
865      PRINT "Rezervor = R";C
870      PRINT "Ziua = ";C$(R)
900      END
```

BASIC-80

```
5     DIM C$(6),B(6),D(3),A(6,3)
10    FOR I=1 TO 6
15       READ C$(I)
20    NEXT I
25    DATA LUNI,MARTI,MIERCURI,JOI,VINERI,SIMBATA
30    PRINT TAB(7);"Introduceti livrarile"
40    PRINT TAB(7);"de benzina pe zile in ordinea"
45    PRINT TAB(7);"       R1     R2      R3    "
50    FOR Z=1 TO 6
51       PRINT C$(Z);" :"
55       S=0
60       FOR I=1 TO 3
61          PRINT "R";: PRINT USING "#";I;:PRINT " ";:INPUT A(Z,I)
65          S=S+A(Z,I)
70       NEXT I
75       B(Z)=S/3
80    NEXT Z
85    FOR I=1 TO 3
90       S=0
95       FOR Z=1 TO 6
100         S=S+A(Z,I)
105      NEXT Z
110      D(I)=S/6
115   NEXT I
120   PRINT "ZIUA","R1","R2","R3","MEDIA"
125   FOR L=1 TO 70
130      PRINT "-";
135   NEXT L
136   PRINT
140   FOR Z=1 TO 6
159      PRINT C$(Z),A(Z,1),A(Z,2),A(Z,3),B(Z)
```

```
160   NEXT Z
161   FOR M=1 TO 70
162      PRINT "-";
163   NEXT M:PRINT
164   PRINT "MEDIA ";
165   FOR I=1 TO 3
167      PRINT D(I),
170   NEXT I
171   PRINT
175   H=A(1,1)
180   R=1
185   C=1
190   FOR Z=1 TO 6
195      FOR I=1 TO 3
200         IF A(Z,I)<=H THEN 220
205         H=A(Z,I)
210         R=Z
215         C=I
220      NEXT I
225   NEXT Z
230   PRINT "Maximul livrarilor ";H;" tone "
235   PRINT "Rezervor = R";C
240   PRINT "Ziua = ";C$(R)
245   END
```

BASIC-PLUS

```
100   DIM C$(6),B(6),D(3),A(6,3)
110   FOR I=1 TO 6
120      READ C$(I)
130   NEXT I
140   DATA LUNI,MARTI,MIERCURI,JOI,VINERI,SIMBATA
150   PRINT TAB(7);"Introduceti livrarile"
160   PRINT TAB(7);"de benzina pe zile in ordinea"
170   PRINT TAB(7);"       R1     R2      R3    "
180   FOR Z=1 TO 6
190      PRINT C$(Z);" :"
200      S=0
210      INPUT "Rezervorul 1";A(Z,1)
220      S=S+A(Z,1)
230      INPUT "Rezervorul 2";A(Z,2)
235      S=S+A(Z,2)
240      INPUT "Rezervorul 3";A(Z,3)
260      S=S+A(Z,3)
270      PRINT
```

```
280       B(Z)=S/3
290    NEXT Z
300    FOR I=1 TO 3
310       PRINT
320       S=0
330       FOR Z=1 TO 6
340          S=S+A(Z,I)
350       NEXT Z
360       D(I)=S/6
370    NEXT I
380    PRINT "ZIUA    "," R1"," R2"," R3"," MEDIA"
390    FOR L=1 TO 69
400       PRINT "-";
410    NEXT L
420    PRINT "-":PRINT
430    FOR Z=1 TO 6
440       PRINT C$(Z),A(Z,1),A(Z,2),A(Z,3),B(Z)
450    NEXT Z
460    FOR M=1 TO 69:PRINT "-";:NEXT M
470    PRINT "-":PRINT
480    PRINT "MEDIA              ";
490    FOR I=1 TO 3
500       PRINT D(I),
510    NEXT I
520    PRINT
530    H=A(1,1)
540    R=1
550    C=1
560    FOR Z=1 TO 6
570       FOR I=1 TO 3
580          IF A(Z,I)<=H THEN 620
590          H=A(Z,I)
600          R=Z
610          C=I
620       NEXT I
630    NEXT Z
640    PRINT
650    PRINT
660    PRINT "Maximul livrarilor ";H;" tone "
670    PRINT "la rezervorul  =";C
680    PRINT "in ziua de  = ";C$(R)
690    PRINT
700    PRINT
710    STOP
720    END
```

```
ABASIC

100   DIM C$(6),B(6),D(3),A(6,3)
110   FOR I=1 TO 6
120      READ C$(I)
130   NEXT I
140   DATA "LUNI","MARTI","MIERCURI","JOI","VINERI","SIMBATA"
150   PRINT TAB(7);"Introduceti livrarile"
160   PRINT TAB(7);"de benzina pe zile in ordinea"
170   PRINT TAB(7);"          R1     R2        R3 "
180   FOR Z=1 TO 6
190      PRINT C$(Z);" :"
200      S=0
210      INPUT "Rezervorul 1";A(Z,1)
220      S=S+A(Z,1)
230      INPUT "Rezervorul 2";A(Z,2)
235      S=S+A(Z,2)
240      INPUT "Rezervorul 3";A(Z,3)
260      S=S+A(Z,3)
270      PRINT
280      B(Z)=S/3
290   NEXT Z
300   FOR I=1 TO 3
310      PRINT
320      S=0
330      FOR Z=1 TO 6
340         S=S+A(Z,I)
350      NEXT Z
360      D(I)=S/6
370   NEXT I
380   PRINT "ZIUA","R1","R2","R3","MEDIA"
390   FOR L=1 TO 69
400      PRINT "-";
410   NEXT L
420   PRINT "-":PRINT
430   FOR Z=1 TO 6
440      PRINT C$(Z),A(Z,1),A(Z,2),A(Z,3),B(Z)
450   NEXT Z
460   FOR M=1 TO 69:PRINT "-";:NEXT M
470   PRINT "-":PRINT
480   PRINT "MEDIA ";
490   FOR I=1 TO 3
500      PRINT D(I),
510   NEXT I
520   PRINT
530   H=A(1,1)
540   R=1
```

```
550   C=1
560   FOR Z=1 TO 6
570     FOR I=1 TO 3
580       IF A(Z,I)<=H THEN 620
590       H=A(Z,I)
600       R=Z
610       C=I
620     NEXT I
630   NEXT Z
640   PRINT
650   PRINT
660   PRINT "Maximul livrarilor ";H;" tone "
670   PRINT "la rezervorul  =";C
680   PRINT "in ziua de = ";C$(R)
690   PRINT
700    PRINT
710    STOP
720    END
```

EXEMPLELE 7

BASIC-AMSTRAD

```
10    REM ** Creare fisier LIVRARI.DAT **
20    PRINT
30    OPENOUT "LIVRARI.DAT"
40    PRINT "Va rog introduceti cod beneficiar/cantitate/pret"
50    PRINT
60    INPUT "Cod,cantitate,pret:";COD,CANT,PRET
70    WHILE COD<>99
80       WRITE #9, COD, CANT, PRET
90       PRINT
100      INPUT "Cod,cantitate,pret:";COD,CANT,PRET
110   WEND
120   CLOSEOUT
130   PRINT "Fisierul LIVRARI.DAT a fost creat"
140   END
```

BASIC-COMMODORE

```
10    REM ** Creare fisier LIVRARI.DAT **
12    A$=CHR$(13)
15    OPEN 15,8,15
```

```
20    OPEN 2,8,2,"LIVRARI.DAT,S,W"
30    PRINT "Va rog introduceti cod benef
iciar/cantitate/pret"
40    PRINT
50    INPUT "Cod:";K%
60    IF K% >99 GOTO 50
70    IF K% =99 GOTO 120
80    INPUT "Cantitate:";C2
90    INPUT "Pret:";C3
100   PRINT #2,K%;A$;C2;A$;C3
110   GOTO 50
120   CLOSE 2 : CLOSE 15
130   PRINT "Fisierul LIVRARI.DAT a fost
creat"
140   END
```

BASIC-80

```
10    REM ** Creare fisier LIVRARI.DAT **
20    PRINT
30    OPEN "O",#1,"LIVRARI.DAT"
40    PRINT "Va rog introduceti : cod beneficiar/cantitate/pret"
50    PRINT
60    INPUT "Cod,cantitate,pret:";COD,CANT,PRET
70    WHILE COD<>99
80       PRINT #1,COD;",";CANT;",";PRET
90       PRINT
100      INPUT "Cod,cantitate,pret:";COD,CANT,PRET
110   WEND
120   CLOSE #1
130   PRINT "Fisierul LIVRARI.DAT a fost creat"
140   END
```

BASIC-PLUS

```
10    REM ** Creare fisier secvential LIVRARI.DAT **
20    OPEN "LIVRARI.DAT" FOR OUTPUT AS ASCII   FILE 1 TO WRITE
30    PRINT "Va rog introduceti cod beneficiar/cantitate/pret"
40    PRINT
50    INPUT "Cod:";K%
60    IF K% >99 GO TO 50
70    IF K% =99 GO TO 120
80    INPUT "Cantitate:";C2
90    INPUT "Pret:";C3
```

```
100   PRINT #1,K%;",";C2;",";C3
110   GOTO 50
120   CLOSE 1
130   PRINT "Fisierul LIVRARI.DAT a fost creat"
140   END
```

ABASIC

```
10    REM ** Creare fisier LIVRARI.DAT **
20    PRINT
30    OPEN #1,"LIVRARI.DAT"
40    PRINT "Va rog introduceti : cod beneficiar/cantitate/pret"
50    PRINT
60    INPUT "Cod,cantitate,pret:";COD,CANT,PRET
70    WHILE COD<>99
80      WRITE #1,COD,CANT,PRET
90      PRINT
100     INPUT "Cod,cantitate,pret:";COD,CANT,PRET
110   WEND
120   CLOSE #1
130   PRINT "Fisierul LIVRARI.DAT a fost creat"
140   END
```

EXEMPLELE 8

BASIC-AMSTRAD

```
10      REM " ** Intretinere fisier LIVRARI.DAT ** "
15      DIM COD(100),CANT(100),PRET(100)
20      GOSUB 1000
30      END
1000      OPTIUNE$=""
1010      FISINCARC=0
1020      WHILE OPTIUNE$<>"6"
1030        PRINT
1040        PRINT "Alegeti 1-6 "
1050        PRINT
1060        PRINT "   1.Incarca fisier LIVRARI.DAT "
1070        PRINT "   2.Salveaza fisier "
1080        PRINT "   3.Cautare in fisier "
1090        PRINT "   4.Actualizare fisier "
1100        PRINT "   5.Listare fisier "
1110        PRINT "   6.End"
1120        PRINT "Va rog optiunea Dvs."
1130        WHILE OPTIUNE$=""
1140          OPTIUNE$=INKEY$
```

```
1150        WEND
1160        IF OPTIUNE$="1" THEN GOSUB 2000
1170        IF OPTIUNE$="2" THEN GOSUB 3000
1180        IF OPTIUNE$="3" THEN GOSUB 4000
1190        IF OPTIUNE$="4" THEN GOSUB 5000
1200        IF OPTIUNE$="5" THEN GOSUB 6000
1210        IF OPTIUNE$<>" " THEN CLS
1220        IF OPTIUNE$<>"6" THEN OPTIUNE$=""
1230    WEND
1240    RETURN
2000    REM " * 1.Incarca fisier LIVRARI.DAT "
2005    CLS
2010    I=1
2020    IF FISINCARC=1 THEN ERASE COD,CANT,PRET
2040    OPENIN "LIVRARI.DAT"
2050    WHILE NOT EOF
2060        INPUT#9,COD(I),CANT(I),PRET(I)
2070        I=I+1
2080    WEND
2090    CLOSEIN
2100    PRINT
2110    PRINT " ==Fisier incarcat == "
2120    INPUT " ( tastati CR ) ";CR
2130    FISINCARC=1
2140    RETURN
3000    REM " * 2.Salveaza fisier "
3005    CLS
3010    OPENOUT "LIVRARI.S"
3020    C=1
3030    WHILE C<I
3040       WRITE #9, COD(C),CANT(C),PRET(C)
3050       C=C+1
3060    WEND
3070    CLOSEOUT
3080    PRINT " == Salvat fisier =="
3090    INPUT " (tastati CR) ";CR
3100    RETURN
4000    REM " *3.Cautare in fisier "
4010    CLS
4020    INPUT "Precizati cod beneficiar ";CBEN
4030    C=1
4035    OPENIN "LIVRARI.DAT"
4040    GASIT=0
4050    WHILE C<I AND GASIT=0
4052       INPUT#9,COD(C),CANT(C),PRET(C)
4054       IF CBEN <>COD(C) THEN 4060
4056       INPUT GASIT=1
```

```
4060     C=C+1
4070   WEND
4072   CLOSEIN
4075   ON GASIT+1 GOTO 4080,4092
4078   REM "Beneficiarul a fost sau nu gasit in fisier"
4080   PRINT " Beneficiarul ";CBEN;"nu este in fisier"
4090   PRINT
4091   GOTO 4100
4092   PRINT "Beneficiarul cu codul ";CBEN;"se afla in fisier "
4100   INPUT " (tastati CR ) ";CR
4110   RETURN
5000   REM " * 4.Actualizare fisier "
5005   CLS
5010   INPUT " Pentru ce cod doriti actualizarea ";CODB
5020   C=1
5030   GASIT=0
5040   WHILE C<I AND GASIT=0
5050     IF CODB<>COD (C) THEN 5060
5051     INPUT " Care este noua valoare a cantitatii livrate ";CANT(C)
5052     GASIT=1
5060     C=C+1
5070   WEND
5080   IF GASIT=0 THEN PRINT " Beneficiarul ";CODB;" nu este in fisier"
5090   PRINT
5100   INPUT " (tasteaza CR) ";CR
5110   RETURN
6000   REM " * 5.Listare fisier "
6010   CLS
6015   PRINT " Cod  Cantitate Pret"
6020   I=0
6030   OPENIN "LIVRARI.DAT"
6050   WHILE NOT EOF
6055     I=I+1
6060     PRINT COD (I);" ";CANT(I);" ";PRET(I)
6080     INPUT#9,COD(I),CANT(I),PRET(I)
6090   WEND
6095   CLOSEIN
6100   INPUT " ( tastati CR ) ";CR
6110   RETURN
```

BASIC-80

```
10     REM " ** Intretinere fisier LIVRARI.DAT ** "
15     DIM COD(100),CANT(100),PRET(100)
20     GOSUB 1000
30     END
```

```
1000    OPTIUNE$=""
1010    FISINCARC=0
1020    WHILE OPTIUNE$<>"6"
1030       PRINT
1040       PRINT "Alegeti 1-6 "
1050       PRINT
1060       PRINT "    1.Incarca fisier LIVRARI.DAT "
1070       PRINT "    2.Salveaza fisier "
1080       PRINT "    3.Cautare in fisier "
1090       PRINT "    4.Actualizare fisier "
1100       PRINT "    5.Listare fisier "
1110       PRINT "    6.End"
1120       PRINT "Va rog optiunea Dvs."
1130       WHILE OPTIUNE$=""
1140         OPTIUNE$=INKEY$
1150       WEND
1160       IF OPTIUNE$="1" THEN GOSUB 2000
1170       IF OPTIUNE$="2" THEN GOSUB 3000
1180       IF OPTIUNE$="3" THEN GOSUB 4000
1190       IF OPTIUNE$="4" THEN GOSUB 5000
1200       IF OPTIUNE$="5" THEN GOSUB 6000
1210       IF OPTIUNE$<>" " THEN PRINT CHR$(24)
1220       IF OPTIUNE$<>"6" THEN OPTIUNE$=""
1230    WEND
1240    RETURN
2000    REM " * 1.Incarca fisier LIVRARI.DAT "
2005    PRINT CHR$(24)
2010    I=1
2020    IF FISINCARC=1 THEN ERASE COD,CANT,PRET
2040    OPEN "I",#1,"LIVRARI.DAT"
2050    WHILE NOT EOF (1)
2060      INPUT#1,COD(I),CANT(I),PRET(I)
2070      I=I+1
2080    WEND
2090    CLOSE#1
2100    PRINT
2110    PRINT " ==Fisier incarcat == "
2120    INPUT " ( tastati CR ) ";CR
2130    FISINCARC=1
2140    RETURN
3000    REM " * 2.Salveaza fisier "
3005    PRINT CHR$(24)
3010    OPEN "O",#2,"LIVRARI.S"
3020    C=1
3030    WHILE C<I
3040      PRINT #2, COD(C),CANT(C),PRET(C)
3050      C=C+1
```

```
3060    WEND
3070    CLOSE#2
3080    PRINT " == Salvat fisier == "
3090    INPUT " (tastati CR) ";CR
3100    RETURN
4000    REM " *3.Cautare in fisier "
4010    PRINT  CHR$(24)
4020    INPUT "Precizati cod beneficiar ";CBEN
4030    C=1
4035    OPEN "I",#1,"LIVRARI.DAT"
4040    GASIT=0
4050    WHILE C<I AND GASIT=0
4052      INPUT#1,COD(C),CANT(C),PRET(C)
4054      IF CBEN <>COD(C) THEN 4060
4056      GASIT=1
4060      C=C+1
4070    WEND
4072    CLOSE#1
4075    ON GASIT+1 GOTO 4080,4092
4078    REM "Beneficiarul a fost sau nu gasit in fisier"
4080    PRINT " Beneficiarul ";CBEN;"nu este in fisier"
4090    PRINT
4091    GOTO 4100
4092    PRINT "Beneficiarul cu codul ";CBEN;"se afla in fisier "
4100    INPUT " (tastati CR ) ";CR
4110    RETURN
5000    REM " * 4.Actualizare fisier "
5005    PRINT CHR$(24)
5010    INPUT " Pentru ce cod doriti actualizarea ";CODB
5020    C=1
5030    GASIT=0
5040    WHILE C<I AND GASIT=0
5050      IF CODB<>COD (C) THEN 5060
5051      INPUT " Care este noua valoare a cantitatii livrate ";CANT(C)
5052      GASIT=1
5060      C=C+1
5070    WEND
5080    IF GASIT=0 THEN PRINT " Beneficiarul ";CODB;" nu este in fisier"
5090    PRINT
5100    INPUT " (tasteaza CR) ";CR
5110    RETURN
6000    REM " * 5.Listare fisier "
6010    PRINT CHR$(24)
6015    LPRINT " Cod  Cantitate Pret"
6020    I=0
6030    OPEN "I",#1, "LIVRARI.DAT"
6050    WHILE NOT EOF(1)
```

```
6055    I=I+1
6060    PRINT COD (I);" ";CANT(I);" ";PRET(I)
6070    LPRINT COD(I);"      ";CANT(I);"     ";PRET(I)
6080    INPUT#1,COD(I),CANT(I),PRET(I)
6090  WEND
6095  CLOSE#1
6100  INPUT " ( tastati CR ) ";CR
6110  RETURN
```

EXEMPLUL 9

BASIC-PLUS

```
100  REM ** INTRETINERE FISIER LIVRARI.DAT **
110  DIM # 1 ,C(100),P(100),Q(100)
120  GOSUB 140
130  END
140  REM SUBRUTINA DE PREZENTARE A MENIULUI
150  PRINT
160  PRINT " ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NUMAR INTRE 1-6 "
170  PRINT
180  PRINT "  1.INCARCA FISIER LIVRARI.DAT "
190  PRINT "  2.ADAUGARE DE BENEFICIARI NOI "
200  PRINT "  3.CAUTARE IN FISIER "
210  PRINT "  4.ACTUALIZARE FISIER "
220  PRINT "  5.LISTARE FISIER "
230  PRINT "  6.END "
240  INPUT " VA ROG SPECIFICATI OPTIUNEA DVS. ",O%
250  IF O% < 7 THEN 270
260  PRINT " *** OPTIUNE ERONATA*** ": GO TO 150
270  IF O%=1 THEN GOSUB 350
280  IF O%=2 THEN GOSUB 510
290  IF O%=3 THEN GOSUB 680
300  IF O%=4 THEN GOSUB 820
310  IF O%=5 THEN GOSUB 1000
320  GO TO 340
330
340  RETURN
350  REM * 1.INCARCA FISIER LIVRARI.DAT
360  OPEN "LIVRARI.DAT" AS VIRTUAL FILE 1
370  FOR K=1 TO 100
380     C(K) = 0 : Q (K) = O :P(K) = O
390  NEXT K
400   INPUT "COD : ";K%
```

```
410   IF K% > 99 GO TO 400
420   IF K%=99 GO TO 470
430   INPUT "CANTITATE : ";C2
440   INPUT "PRET      : ";C3
450   C(K%) = K% : Q(K%) =C2 : P(K%) =C3
460   GO TO 400
470   CLOSE 1
480   PRINT
490   PRINT " ==FISIER INCARCAT == "
500   RETURN
510   REM * 2.ADAUGARI DE NOI INREGISTRARI
520   OPEN "LIVRARI.DAT" AS VIRTUAL FILE 1
530   INPUT "PRECIZATI CODUL BENEFICIARULUI : ";K%
540   IF K%< 99 GO TO 570
550   PRINT "COD BENEFICIAR ERONAT"
560   GO TO 530
570   IF C(K%) = 0 THEN 600
580   PRINT "BENEFICIAR EXISTENT - OPERATIE NEPERMISA"
590   GO TO 650
600   INPUT "CANTITATE : ";C2
610   INPUT "PRET      : ";C3
620   C(K%) = K% : Q(K%) = C2 : P(K%) = C3
630
640
650   CLOSE 1
660   PRINT " == TERMINAT ADAUGARE == "
670   RETURN
680   REM * 3.CAUTARE IN FISIER
690   OPEN "LIVRARI.DAT" AS VIRTUAL FILE 1
700   INPUT "PRECIZATI CODUL BENEFICIARULUI : ";K%
710   IF K% < 99 GO TO 740
720   PRINT "COD BENEFICIAR ERONAT"
730   GO TO 700
740   A$ = "NU A"
750   IF C(K%) = 0 THEN 770
760   A$ = "A"
770   CLOSE 1
780   REM BENEFICIARUL A FOST/NU A FOST GASIT IN FISIER
790   PRINT
800   PRINT " BENEFICIARUL CU CODUL ";K%;A$;" FOST GASIT IN FISIER "
810   RETURN
820   REM * 4.ACTUALIZARE FISIER "
830   INPUT " PENTRU CE BENEFICIAR DORITI ACTUALIZAREA ";K%
840   IF K% < 99 GO TO 870
850   PRINT "COD BENEFICIAR ERONAT"
860   GO TO 830
870   OPEN "LIVRARI.DAT" AS VIRTUAL FILE 1
```

```
880   IF C(K%) = 0 THEN 970
890   G% = 1
900   PRINT "LA BENEF.:";K%;"ESTE CANT.:";Q(K%);"SI PRETUL :";P(K%)
910   PRINT "SPECIFICATI NOILE VALORI : "
920   INPUT "                         CANTITATE";C2
930   INPUT "                         PRET      ";C3
940   Q(K%) = C2 : P(K%) = C3
950
960   IF G% = 1 THEN 980
970   PRINT "BENEFICIARUL";K%;"NU ESTE PREZENT IN FISIER"
980   CLOSE 1
990   RETURN
1000  REM * 5.LISTARE FISIER
1010  OPEN "LIVRARI.DAT" AS VIRTUAL FILE 1
1020  PRINT "COD BENEF.";TAB (30);"CANTITATE";TAB(50);"PRET"
1030  FOR I =1 TO 100
1040    IF C(I) = 0 THEN 1060
1050    PRINT C(I);TAB(30);Q(I);TAB(50);P(I)
1060  NEXT I
1070  CLOSE 1
1080  PRINT:PRINT "LISTARE TERMINATA"
1090  RETURN
```

EXEMPLUL 10

BASIC–PLUS

```
100    REM PROGRAMUL CREAZA SI INTRETINE UN FISIER DE BENEFICIARI
110    REM FUNCTIUNILE PROGRAMULUI :
120    REM -CREARE FISIER BENEFICIARI
130    REM -ADAUGARE BENEFICIARI
140    REM -CONSULTARE FISIER BENEFICIARI
150    REM -STERGERE BENEFICIARI
160    REM -LISTARE BENEFICIARI
170    REM -MODIFICARE BENEFICIAR EXISTENT
180    REM DEFINIREA FISIERULUI VIRTUAL
190    DIM #1,B$(30)=40,C(30),P(30),B1$(30,30)
200    REM B$ -NUME BENEFICIAR
210    REM C- CANTITATE
220    REM P- PRET
230    PRINT
240    PRINT " 1. CREARE FISIER BENEFICIARI "
250    PRINT " 2. ADAUGARE BENEFICIARI "
260    PRINT " 3. CONSULTARE FISIER BENEFICIARI "
270    PRINT " 4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER "
280    PRINT " 5. STERGERE DIN FISIER BENEFICIARI "
290    PRINT " 6. MODIFICARE INREGISTRARI BENEFICIAR "
```

```
300  PRINT " 7. SFIRSIT  SESIUNE ACTUALA "
310  PRINT
320  INPUT " PRECIZATI OPTIUNEA DVS.";O
330  IF O < 8 THEN 360
340  PRINT "OPTIUNE ERONATA "
350  GO TO 230
360  IF O = 1 THEN GOSUB 450
370  IF O = 2 THEN GOSUB 570
380  IF O = 3 THEN GOSUB 800
390  IF O = 4 THEN GOSUB 1350
400  IF O = 5 THEN GOSUB 950
410  IF O = 6 THEN GOSUB 1130
420  IF O = 7 THEN GOTO 1720
430  G0 TO 230
440  PRINT
450  REM ++++++++++++++++++++++++++++++++++ CREARE FISIER BENEFICIARI
460  IF T1 <> 0 THEN 530
470  GOSUB 1470 ! DESCHIDERE
480  FOR I=1
490     B$(I)="":P(I)=0:C(I)=0
500     NEXT I
510  T1=T1+1
520  GO TO 550
530  PRINT "FISIERUL EXISTA"
540  GO TO 340
550  GOSUB 1530 !INCHIDERE
560  RETURN
570  REM +++++++++++++++++++++++++++++++++++ ADAUGARE BENEFICIARI NOI
580  T1=T1+1
590  GOSUB 1630 ! NUMARARE ARTICOLE
600  GOSUB 1470 ! DESCHIDERE
610  IF I3 =30 THEN 710
620  INPUT "NUMELE BENEFICIARULUI :",N$
630  T=0
640  A=31
650  FOR I=30 TO 1 STEP-1
660     IF C(I)=0 THEN A=I ! RETIN PRIMA CASUTA GOALA
670     IF N$=B$(I) THEN T=1 ! BENEFICIAR EXISTENT IN FISIER
680  NEXT I
690  IF T=1 THEN 770
700  IF A < 31 THEN 730
710  PRINT " * FISIERUL ESTE PLIN * ADAUGARILE SE REFUZA * "
720  GO TO 780
730  B$(A)=N$ :I3=I3+1! RETIN NUMELE BENEFICIARULUI
740  INPUT " CANTITATE " ,C(A) !RETIN CANTITATEA
750  INPUT "PRET          ",P(A) !RETIN PRETUL
760  GO TO 780
770  PRINT "BENEFICIARUL  ";N$;" ESTE DEJA PREZENT IN FISIER "
```

```
780    GOSUB 1530 ! INCHIDERE
790    RETURN
800    REM ++++++++++++++++++++++++++++ CONSULTARE FISIER BENEFICIARI
810    T1=T1+1
820    GOSUB 1470 ! DESCHIDERE
830    T = 0
840    INPUT "NUMELE BENEFICIARULUI : ",N$
850    FOR I = 1 TO 30
860       IF N$ = B$(I) THEN T=I ! DETECTARE PREZENTA BENEFICIAR SOLICITAT
870    NEXT I
880    IF T <> 0 THEN 910
890    PRINT "BENEFICIARUL ";N$;" NU EXISTA IN FISIER"
900    GO TO 930
910    GOSUB 1590
920    PRINT B$(T);TAB(21);C(T);TAB(31);P(T)
930    GOSUB 1530 ! INCHIDERE
940    RETURN
950    REM+++++++++++++++++++++++++++ STERGERE BENEFICIAR DIN FISIER
960    T=0
970    T1=T1+1
980    INPUT "NUMELE BENEFICIARULUI : ",N$
990    GOSUB 1470 !DESCHIDERE
1000   FOR I = 1 TO 30
1010      IF N$ <> B$(I) THEN 1060
1020      B$(I) = ""
1030      T=1
1040      P(I)=0
1050      C(I)=0
1060   NEXT I
1070   IF T=1 THEN 1100
1080   PRINT "BENEFICIAR INEXISTENT IN FISIER"
1090   GOTO 1110
1100   PRINT "A FOST STERS BENEFICIARUL ";N$
1110   GOSUB 1530 ! INCHIDERE
1120   RETURN
1130   REM ++++++++++++++++++++++++ MODIFICARE INREGISTRARE BENEFICIAR
1140   GO SUB 1470 ! DESCHIDERE
1150   T = 0:T1=T1+1
1160   INPUT "NUMELE BENEFICIARULUI : ",N$
1170   FOR I = 1 TO 30
1180      IF N$ = B$(I) THEN T=I ! DETECTARE PREZENTA BENEFICIAR SOLICITAT
1190   NEXT I
1200   IF T <> 0 THEN 1230
1210   PRINT "BENEFICIARUL ";N$;" NU EXISTA IN FISIER"
1220   GO TO 1330
1230   PRINT "SPECIFICATI MODIFICARILE DORITE "
1240   INPUT "NUME BENEFICIAR ",M$
1250   INPUT "CANTITATE        ",C1
```

```
1260    INPUT "PRET          ",P1
1270    IF M$ <> "" THEN B$(T) = M$
1280    IF C1 <> 0 THEN C(T) = C1
1290    IF P1 <> 0 THEN P(T) = P1
1300    PRINT"NOUA INREGISTRARE :"
1310    GOSUB 1590
1320    PRINT B$(T);TAB(21);C(T);TAB(31);P(T)
1330    GO SUB 1530 ! INCHIDERE
1340    RETURN
1350    REM SRT DE NUMARARE A INREG EXISTENTE IN FISIER
1360    T1=T1+1
1370    GOSUB 1630 ! NUMARARE
1380    PRINT:PRINT
1390    PRINT "FISIERUL CONTINE ";I3;"INREGISTRARI"
1400    GOSUB 1470
1410    GOSUB 1590
1420    PRINT B$(P9),C(P9),P(P9) FOR P9 =1 TO I3
1430    PRINT
1440    GOSUB 1530
1450    RETURN
1460    REM SUBRUTINE
1470    REM DESCHIDERE FISIER
1480    OPEN "BENEF.DAT" AS VIRTUAL FILE 1
1490    PRINT
1500    PRINT "  ****   INCEPUT FAZA   ****   "
1510    PRINT
1520    RETURN
1530    REM INCHIDERE FISIER
1540    CLOSE 1
1550    PRINT
1560    PRINT "    ****    TERMINAT FAZA    ****  "
1570    PRINT
1580    RETURN
1590    REM CAP TABEL
1600    PRINT "BENEFICIAR /CANTITATE LIVRATA/ PRET"
1610    PRINT "===================================":PRINT
1620    RETURN
1630    REM SUBRUTINA NUMARARE BENEFICIARI INSCRISI IN FISIER
1640    OPEN "BENEF.DAT" AS VIRTUAL FILE 1
1650    I3=0
1660    FOR I=1 TO 30
1670       IF B$(I)="" THEN 1690
1680       I3=I3+1
1690    NEXT I
1700    CLOSE 1
1710    RETURN
1720    STOP
1730    END
```

EXEMPLELE 11

BASIC-AMSTRAD

```
140   DIM ben$(30),cant(30),pret(30),s9(30,3)
150   PRINT
160   PRINT" 1.Adaugare beneficiari "
170   PRINT" 2.Listare beneficiari "
180   PRINT" 3.Consultare beneficiari"
190   PRINT" 4.Sortare date "
200   PRINT" 5.Salvare fisier beneficiari "
210   PRINT" 6.Incarcare fisier beneficiari "
220   PRINT"  (0 pentru END)"
230   PRINT
240   INPUT " Precizati optiunea Dvs.";OPT
250   REM "sterge ecran"
260   CLS
270   ON opt GOSUB 330,460,540,850,640,730
280   IF opt=0 THEN END
290   PRINT "Apasati o tasta pentru reintoarcerea in meniu"
300   IF INKEY$="" THEN 300
310   GOTO 150
320   PRINT
330   REM"Adaugare beneficiari"
340   iii=0
350   FOR i=1 TO 30
360      CLS
370      IF LEN(ben$(i))>0 THEN 430
380      INPUT "Nume beneficiar ";ben$(i)
390      IF ben$(i)="" THEN 440
400      iii=iii+1
410      INPUT "Livrari ";cant(i)
420      INPUT "Pret tona ";pret (i)
430   NEXT i
440   PRINT "Terminat introducere date"
450   RETURN
460   REM"Listare beneficiari"
470   PRINT
480   FOR i=1 TO 30
490      IF ben$(i)= "" THEN 520
500      PRINT ben$(i);"";cant(i);"";pret(i)
510   NEXT i
520   PRINT "Terminat listare fisier"
530   RETURN
540   REM "Consultare fisier beneficiari"
550   PRINT
```

```
560    INPUT " Ce beneficiar cautati ";nben$
570    FOR i=1 TO 30
580      IF INSTR(ben$(i),nben$)=0 THEN 610
590      PRINT ben$(i);" ";cant(i);" ";pret(i)
600      RETURN
610    NEXT i
620    PRINT "Beneficiarui";nben$;"nu este in fisier"
630    RETURN
640    REM "Salvare fisier beneficiari"
650    PRINT
660    OPENOUT "benef.dat"
670    FOR i=1 TO 30
680      WRITE #9,ben$(i),cant(i),pret(i)
690    NEXT i
700    CLOSEOUT
710    PRINT "== Fisier salvat =="
720    RETURN
730    REM "Citire fisier beneficiari"
740    iii=0
750    PRINT
760    OPENIN "benef.dat"
770    FOR i=1 TO 30
780      INPUT #9,ben$(i),cant(i),pret(i)
790      IF ben$(i)="" THEN 820
800      iii=iii+1
810    NEXT i
820    CLOSEIN
830    PRINT "Fisier incarcat"
840    RETURN
850    REM "Program de sortare date"
860    GOSUB 1500
870    GOSUB 1340
880    RETURN
890    REM "bubble demo"
900    n=iii
910    PRINT "Lista beneficiari (nesortata)"
920    FOR i=1 TO n
930      PRINT ben$(i),cant(i),pret(i)
940    NEXT i
950    PRINT:PRINT
960    FOR b=1 TO n-1
970      PRINT "Pasul -->";b
980      f=0
990      FOR c=1 TO n-b
1000       FOR d=1 TO n
1010         b$(c,d)=ben$(d)
1020       NEXT d
```

```
1030            IF ben$(c+1)>=ben$(c) THEN 1080
1040            v$=ben$(c)
1041            ben$(c)=ben$(c+1)
1042            ben$(c+1)=v$
1050            z=cant(c)
1051            cant(c)=cant(c+1)
1052            cant(c+1)=z
1060            n=pret(c)
1061            pret(c)=pret(c+1)
1062            pret(c+1)=n
1070            f=1
1080        NEXT c
1090        FOR d=1 TO n
1100          b$(n-b+1,d)=ben$(d)
1110        NEXT d
1120        FOR d=1 TO n
1130            IF d=1 THEN PRINT " * ";:d1=-1:GOTO 1210
1140            d1=d-2
1150            IF d1>=n-b THEN d1=n-b+1
1160            IF d1=0 THEN 1180
1170            FOR e=1 TO d1
1172                PRINT TAB (e*6-3);b$(e,d)
1175            NEXT e
1180            IF d>n-b+1 THEN 1220
1190            PRINT TAB(6*d1+2);"*";b$(d1+2,d);:
1200            PRINT TAB (6*d1+8);"*";
1210            FOR e=d1+2 TO n-b+1:PRINT TAB(e*6-3);b$(e,d);:NEXT e
1220            IF d<>n-b+1 THEN 1250
1230            PRINT
1240            FOR e=1 TO 6*(n-b+1):PRINT "-";:NEXT e
1250            PRINT
1260        NEXT d
1270        IF f=0 THEN PRINT:PRINT " ==Terminat== ":GOTO 1290
1280        PRINT:NEXT b
1290        PRINT:PRINT "Lista beneficiari(sortata)"
1300        FOR i=1 TO n
1310            PRINT ben$(i),cant(i),pret(i)
1320        NEXT i
1330        RETURN
1340        PRINT:PRINT "Apasati o tasta"
1345        IF INKEY$="" GOTO 1345
1350        CLS
1360      PRINT TAB(5);"Metoda de sortare:"
1370      PRINT TAB(7);"1.BUBBLE DEMO"
1380      PRINT TAB(7);"2.BUBBLE SORT"
1390      PRINT TAB(7);"3.Sortare prin extractie"
1400      PRINT TAB(7);"4.Sortare prin insertie"
```

```
1410    PRINT TAB(7);"5.Sortare indexata"
1420    PRINT TAB(7);"6.SHELL"
1430    PRINT TAB(7);"7.QUICK SORT (versiunea 1)"
1440    PRINT TAB(7);"8.QUICK SORT (varianta structurata)"
1450    PRINT TAB(7);"(O pentru END)"
1460    PRINT:PRINT TAB(5);: INPUT "Metoda (1-8) ";met
1470    ON met GOSUB 890,1550,1730,1940,2180,2520,2870,3200
1480    IF met<>0 THEN 1340
1490    RETURN
1500    REM
1510    PRINT" Lista nesortata : ":PRINT
1520    PRINT "Beneficiari/canttitati livrate/pret":PRINT
1530    FOR b=1 TO iii:PRINT ben$(b);TAB(21);cant(b);TAB(31);pret(b):NEXT b
1540    RETURN
1550    REM "Sortare BUBBLE SORT"
1560    i=iii
1570    REM
1580    FOR m=1 TO i-1
1590       f=0
1600       FOR n=1 TO i-m
1610         IF ben$(n+1)>=ben$(n) THEN 1660
1620         v$=ben$(n)
1621         ben$(n)=ben$(n+1)
1622         ben$(n+1)=v$
1630         z=cant(n)
1631         cant(n)=cant(n+1)
1632         cant(n+1)=z
1640         n=pret(n)
1641         pret(n)=pret(n+1)
1642         pret(n+1)=n
1650         f=1
1660       NEXT n
1670       IF f=0 THEN 1690
1680    NEXT m
1690    PRINT:PRINT "Fisier beneficiari (sortat)"
1700    FOR b=1 TO i:PRINT ben$(b),cant(b),pret(b):NEXT b
1710    PRINT
1720    RETURN
1730    REM  "Sortare prin extractie"
1740    i=iii
1750    FOR m=1 TO i-1
1760       REM "Se considera ca ben$(m) este minimul"
1770       MIN$=ben$(m)
1780       j=m
1790       REM
1800       REM
1810       FOR k=m+1 TO i
```

```
1820            IF ben$(k)>= MIN$ THEN 1850
1830            MIN$=ben$(k)
1840            j=m
1850        NEXT k
1860        REM
1870        V$=ben$(m)
1871        ben$(m)=ben$(j)
1872        ben$(j)=v$
1880        z=cant(m)
1881        cant(m)=cant(j)
1882        cant(j)=z
1890        n=pret(m)
1891        pret (m)=pret(j)
1892        pret (j)=n
1900     NEXT m
1910     PRINT "Lista sortata :"
1920     FOR b=1 TO i:PRINT ben$(b),cant(b),pret(b):NEXT b
1930     RETURN
1940     REM "Sortare prin insertie"
1950     stinga=1:dreapta=iii
1960     i=iii
1970     FOR m=stinga+1 TO dreapta
1980        REM "Insertie"
1990        GOSUB 2050
2000     NEXT m
2010     PRINT "Lista sortata:"
2020     FOR b=1 TO i:PRINT ben$(b),cant(b),pret(b):NEXT b
2030     RETURN
2040     REM"Procedura de insertie"
2050     PRINT
2060     d=m-1
2070     u$=ben$(m)
2080     c=cant(m)
2090     p=pret(m)
2100     IF d<stinga OR u$>=ben$(d) THEN 2160
2110     ben$(d+1)=ben$(d)
2120     cant(d+1)=cant(d):pret(d+1)=pret(d)
2130     d=d-1
2140     GOTO 2100
2150     cant(d+1)=c:pret(d+1)=p
2160     ben$(d+1)=u$
2170     RETURN
2180     REM "Sortare indexata"
2190     n=iii
2200     FOR r=1 TO n
2210        n$(r,1)=ben$(r)
2220        r1=cant(r):n$(r,2)=STR$(r1)
```

```
2230        r2=pret(r):n$(r,3)=STR$(r2)
2240     NEXT r
2250     INPUT "Pentru continuare tastati (Y/N) ";a$
2260     WHILE ASC(a$)=89
2270        INPUT "Cimpul : 1.Beneficiar/2.Cantitate/3.Pret(1/2/3) ";j
2280        FOR r=1 TO n
2290        k$(r)=n$(r,j)
2300        NEXT r
2310        GOSUB 2430
2320        PRINT"Inregistrari sortate dupa cimpul ";j
2330        FOR r=1 TO n
2340           FOR i=1 TO 3
2350              rl=x(r):PRINT n$(rl,i);" ",
2360           NEXT i
2370           PRINT
2380        NEXT r
2390        PRINT
2400        INPUT "Pentru continuare tastati (Y/N) ";a$
2410     WEND
2420     RETURN
2430     FOR a=1 TO n
2440        p=1
2450        FOR b=1 TO n
2460           IF k$(a)>k$(b) THEN p=p+1
2470           IF k$(a)=k$(b) AND a>b THEN p=p+1
2480        NEXT b
2490        x(p)=a
2500     NEXT a
2510     RETURN
2520     REM "Sortare prin metoda SHELL"
2530     i=iii:c=0:s=0
2540     PRINT "Lista nesortata"
2550     GOSUB 2620
2560     REM"Sortare lista"
2570     GOSUB 2690
2580     PRINT "Lista sortata"
2590     GOSUB 2620
2600     PRINT c;"Comparatii":PRINT s;"Schimbari"
2610     RETURN
2620     REM "Tiparire lista nesortata"
2630     FOR k=1 TO i
2640        PRINT ben$(k);TAB(10);cant(k);TAB(20);pret(k)
2670     NEXT k
2680     RETURN
2690     REM "Metoda de sortare SHELL"
2700     g=i
2710     WHILE g>1
```

```
2720        g=INT (g/2)
2730        m=i-g
2740        f=0
2750        FOR k=1 TO m
2760          p=k+g
2770          c=c+1
2780          IF ben$(k)<=ben$(p) THEN 2830
2790          s=s+1
2800          v$=ben$(k)
2801          ben$(k)=ben$(p)
2802          ben$(p)=v$
2805          z=cant(k)
2806          cant(k)=cant(p)
2807          cant(p)=z
2808          n=pret(k)
2809          pret(k)=pret(p)
2810          pret(p)=n
2820          f=1
2830        NEXT k
2840        IF f>0 THEN 2740
2850      WEND
2860      RETURN
2870      REM "QUICK SORT(versiunea 1)"
2880      n=iii:PRINT "Lista nesortata"
2890      I=N:GOSUB 2620
2900      C1=0:S1=0
2910      i1=1:j1=n
2920      i=i1;j=j1:s=-1
2930      c1=c1+1
2940      IF ben$(i)<=ben$(j) THEN 2980
2950      s1=s1+1
2960      v$=ben$(i)
2961      ben$(i)=ben$(j)
2962      ben$(j)=v$
2963      z=cant(i)
2964      cant(i)=cant(j)
2965      cant(j)=z
2966      n=pret(i)
2967      pret(i)=pret(j)
2968      pret(j)=n
2970      s=SGN(-s)
2980      IF s=1 THEN i=i+1 ELSE j=j-1
2990      IF i<j THEN 2930
3000      IF i+1>=j1 THEN 3020
3010      p=p+1:s9(p,1)=i+1:s9(p,2)=j1
3020      j1=i-1
3030      IF i1<j1 THEN 2920
```

```
3040    IF p=0 THEN 3070
3050    il=s9(p,1):jl=s9(p,2):p=p-1
3060    GOTO 2920
3070    PRINT:PRINT "Lista sortata"
3080    i=n:GOSUB 2620
3090    PRINT:PRINT n;" beneficiari "
3100    PRINT cl;" comparatii "
3110    PRINT sl;" schimbari "
3120    PRINT
3130    RETURN
3200    REM "QUICK SORT (varianta structurata)"
3210    PRINT "Lista nesortata"
3215    i=iii:GOSUB 2620
3220    n=iii
3230    p=1:s9(1,1)=1:s9(1,2)=n
3240    WHILE P>0
3250        il=s9(p,1):jl=s9(p,2):p=p-1
3260        WHILE il<jl
3270            i=il:j=jl:s%=-1
3280            WHILE i<j
3290                IF ben$(i)<=ben$(j) THEN 3340
3300                v$=ben(i)
3301                ben$(i)=ben$(j)
3302                ben$(j)=v$
3310                z=cant(i)
3311                cant(i)=cant(j)
3312                cant(j)=z
3320                n=pret(i)
3321                pret(i)=pret(j)
3322                pret(j)=n
3330                s%=-(s%+1)
3340                IF NOT s% THEN j=j-1 ELSE i=i+1
3350            WEND
3360            IF i+1>=jl THEN 3400
3370            p=p+1
3380            s9(p,1)=i+1
3390            s9(p,2)=jl
3400            jl=i-1
3410        WEND
3420    WEND
3430    PRINT
3440    PRINT "Lista sortata"
3450    i=n:GOSUB 2620
3460    RETURN
```

BASIC-80

```
100   REM "Programul creeaza un fisier de beneficiari"
110   REM "Functiunile programului: adaugare beneficiari ,"
120   REM "listare beneficiari ,consultare fisier beneficiari ,"
130   REM "salvare fisier beneficiari"
140   DIM ben$(30),cant(30),pret(30),s9(30,3)
150   PRINT
160   PRINT" 1.Adaugare beneficiari "
170   PRINT" 2.Listare beneficiari "
180   PRINT" 3.Consultare beneficiari"
190   PRINT" 4.Sortare date "
200   PRINT" 5.Salvare fisier beneficiari "
210   PRINT" 6.Incarcare fisier beneficiari "
220   PRINT"  (0 pentru END)"
230   PRINT
240   INPUT" Precizati optiunea Dvs.";OPT
250   REM "sterge ecran"
260   PRINT CHR$(24)
270   ON OPT GOSUB 330,460,540,850,640,730
280   IF OPT=0 THEN END
290   PRINT "Apasati o tasta pentru reintoarcerea in meniu"
300   IF INKEY$="" THEN 300
310   GOTO 150
320   PRINT
330   REM"Adaugare beneficiari"
340   iii=0
350   FOR i=1 TO 30
360      PRINT CHR$(24)
370      IF LEN(ben$(i))>0 THEN 430
380      INPUT "Nume beneficiar ";ben$(i)
390      IF ben$(i)=""THEN 440
400      iii=iii+1
410      INPUT "Livrari ";cant(i)
420      INPUT "Pret tona ";pret (i)
430   NEXT i
440   PRINT "Terminat introducere date"
450   RETURN
460   REM"Listare beneficiari"
470   PRINT
480   FOR i=1 TO 30
490      IF ben$(i)="" THEN 520
500      PRINT ben$(i);"";cant(i);"";pret(i)
510   NEXT i
520   PRINT "Terminat listare fisier"
530   RETURN
```

```
540   REM "Consultare fisier beneficiari"
550   PRINT
560   INPUT " Ce beneficiar cautati ";nben$
570   FOR i=1 TO 30
580     IF INSTR(ben$(i),nben$)=0 THEN 610
590     PRINT ben$(i);" ";cant(i);" ";pret(i)
600     RETURN
610   NEXT i
620   PRINT "Beneficiarul";nben$;"nu este in fisier"
630   RETURN
640   REM "Salvare fisier beneficiari"
650   PRINT
660   OPEN "O",#1,"BENEF.DAT"
670   FOR i=1 TO 30
680     WRITE #1,ben$(i),cant(i),pret(i)
690   NEXT i
700   CLOSE #1
710   PRINT "== Fisier salvat =="
720   RETURN
730   REM "Citire fisiere beneficiari"
740   iii=0
750   PRINT
760   OPEN "I",#1,"BENEF.DAT"
770   FOR i=1 TO 30
780     INPUT #1,ben$(i),cant(i),pret(i)
790     IF ben$(I)="" THEN 820
800     iii=iii+1
810   NEXT i
820   CLOSE #1
830   PRINT "Fisier incarcat"
840   RETURN
850   REM "Program de sortare date"
860   GOSUB 1500
870   GOSUB 1340
880   RETURN
890   REM "bubble demo"
900   n=iii
910   PRINT "Lista beneficiari (nesortata)"
920   FOR i=1 TO n
930     PRINT ben$(i),cant(i),pret(i)
940   NEXT i
950   PRINT:PRINT
960   FOR b=1 TO n-1
970     PRINT "Pasul -->";b
980     f=0
990     FOR c=1 TO n-b
1000      FOR d=1 TO n
```

```
1010          b$(c,d)=ben$(d)
1020        NEXT d
1030        IF ben$(c+1)>=ben$(c) THEN 1080
1040        SWAP ben$(c),ben$(c+1)
1050        SWAP cant(c),cant(c+1)
1060        SWAP pret(c),pret(c+1)
1070        f=1
1080      NEXT c
1090      FOR d=1 TO n
1100        b$(n-b+1,d)=ben$(d)
1110      NEXT d
1120      FOR d=1 TO n
1130        IF d=1 THEN PRINT " * ";:d1=-1:GOTO 1210
1140        d1=d-2
1150        IF d1>=n-b THEN d1=n-b+1
1160        IF d1=0 THEN 1180
1170        FOR e=1 TO d1:PRINT TAB (e*6-3);b$(e,d);:NEXT e
1180        IF d>n-b+1 THEN 1220
1190        PRINT TAB(6*d1+2);"*";b$(d1+2,d);:
1200        PRINT TAB (6*d1+8);"*";
1210        FOR e=d1+2 TO n-b+1:PRINT TAB(e*6-3);b$(e,d);:NEXT e
1220        IF d<>n-b+1 THEN 1250
1230        PRINT
1240        FOR e=1 TO 6*(n-b+1):PRINT "-";:NEXT e
1250        PRINT
1260      NEXT d
1270      IF f=0 THEN PRINT:PRINT " ==Terminat== ":GOTO 1290
1280      PRINT:NEXT b
1290      PRINT:PRINT "Lista beneficiari(sortata)"
1300      FOR i=1 TO n
1310        PRINT ben$(i),cant(i),pret(i)
1320      NEXT i
1330      RETURN
1340      PRINT:PRINT "Apasati o tasta";:aaa$=INPUT$(1)
1350      PRINT CHR$(24)
1360    PRINT TAB(5);"Metoda de sortare:"
1370    PRINT TAB(7);"1.BUBBLE DEMO"
1380    PRINT TAB(7);"2.BUBBLE SORT"
1390    PRINT TAB(7);"3.Sortare prin extractie"
1400    PRINT TAB(7);"4.Sortare prin insertie"
1410    PRINT TAB(7);"5.Sortare indexata"
1420    PRINT TAB(7);"6.SHELL"
1430    PRINT TAB(7);"7.QUICK SORT (versiunea 1)"
1440    PRINT TAB(7);"8.QUICK SORT (varianta structurata)"
1450    PRINT TAB(7);"(0 pentru END)"
1460    PRINT:PRINT TAB(5);: INPUT "Metoda (1-8) ";met
1470    ON met GOSUB 890,1550,1730,1940,2180,2520,2870,3200
```

```
1480    IF met<>0 THEN 1340
1490    RETURN
1500    REM
1510    PRINT" Lista nesortata : ":PRINT
1520    PRINT "Beneficiari/canttitati livrate/pret":PRINT
1530    FOR b=1 TO iii:PRINT ben$(b);TAB(21);cant(b);TAB(31);pret(b):NEXT b
1540    RETURN
1550    REM "Sortare BUBBLE SORT"
1560    i=iii
1570    REM
1580    FOR m=1 TO i-1
1590      f=0
1600      FOR n=1 TO i-m
1610        IF ben$(n+1)>=ben$(n) THEN 1660
1620        SWAP ben$(n),ben$(n+1)
1630        SWAP cant(n),cant(n+1)
1640        SWAP pret(n),pret(n+1)
1650        f=1
1660      NEXT n
1670      IF f=0 THEN 1690
1680    NEXT m
1690    PRINT:PRINT "Fisier beneficiari (sortat)"
1700    FOR b=1 TO i:PRINT ben$(b),cant(b),pret(b):NEXT b
1710    PRINT
1720    RETURN
1730    REM  "Sortare prin extractie"
1740    i=iii
1750    FOR m=1 TO i-1
1760      REM "Se considera ca ben$(m) este minimul"
1770      MIN$=ben$(m)
1780      j=m
1790      REM
1800      REM
1810      FOR k=m+1 TO i
1820        IF ben$(k)>= MIN$ THEN 1850
1830        MIN$=ben$(k)
1840        j=m
1850      NEXT k
1860      REM
1870      SWAP ben$(m),ben$(j)
1880      SWAP cant(m),cant(j)
1890      SWAP pret(m),pret(j)
1900    NEXT m
1910    PRINT "Lista sortata :"
1920    FOR b=1 TO i:PRINT ben$(b),cant(b),pret(b):NEXT b
1930    RETURN
1940    REM "Sortare prin insertie"
```

```
1960    i=iii
1970    FOR m=stinga+l TO dreapta
1980       REM "Insertie
1990       GOSUB 2050
2000    NEXT m
2010    PRINT "Lista sortata:"
2020    FOR b=l TO i:PRINT ben$(b),cant(b),pret(b):NEXT b
2030    RETURN
2040    REM"Procedura de insertie"
2050    PRINT
2060    d=m-l
2070    u$=ben$(m)
2080    c=cant(m)
2090    p=pret(m)
2100    IF d<stinga OR u$>=ben$(d) THEN 2160
2110    ben$(d+l)=ben$(d)
2120    cant(d+l)=cant(d):pret(d+l)=pret(d)
2130    d=d-l
2140    GOTO 2100
2150    cant(d+l)=c:pret(d+l)=p
2160    ben$(d+l)=u$
2170    RETURN
2180    REM "Sortare indexata"
2190    n=iii
2200    FOR r=l TO n
2210       n$(r,l)=ben$(r)
2220       rl=cant(r):n$(r,2)=STR$(rl)
2230       r2=pret(r):n$(r,3)=STR$(r2)
2240    NEXT r
2250    INPUT "Pentru continuare tastati (Y/N) ";a$
2260    WHILE ASC(a$)=89
2270       INPUT "Cimpul : l.Beneficiar/2.Cantitate/3.Pret(l/2/3) ";j
2280       FOR r=l TO n
2290          k$(r)=n$(r,j)
2300       NEXT r
2310       GOSUB 2430
2320       PRINT "Inregistrari sortate dupa cimpul ";j
2330       FOR r=l TO n
2340          FOR i=l TO 3
2350             rl=x(r):PRINT n$(rl,i);" ",
2360          NEXT i
2370          PRINT
2380       NEXT r
2390       PRINT
2400       INPUT "Pentru continuare tastati (Y/N) ";a$
2410    WEND
2420    RETURN
```

```
2430    FOR a=1 TO n
2440      p=1
2450      FOR b=1 TO n
2460        IF k$(a)>k$(b) THEN p=p+1
2470        IF k$(a)=k$(b) AND a>b THEN p=p+1
2480      NEXT b
2490      x(p)=a
2500    NEXT a
2510    RETURN
2520    REM "Sortare prin metoda SHELL"
2530    i=iii:c=0:s=0
2540    PRINT "Lista nesortata"
2550    GOSUB 2620
2560    REM"Sortare lista"
2570    GOSUB 2690
2580    PRINT "Lista sortata",
2590    GOSUB 2620
2600    PRINT c;"Comparatii":PRINT s;"Schimbari"
2610    RETURN
2620    REM "Tiparire lista nesortata"
2630    FOR k=1 TO i
2640      PRINT ben$(k);TAB(10);cant(k);TAB(20);pret(k)
2670    NEXT k
2680    RETURN
2690    REM "Metoda de sortare SHELL"
2700    g=i
2710    WHILE g>1
2720      g=INT (g/2)
2730      m=i-g
2740      f=0
2750      FOR k=1 TO m
2760        p=k+g
2770        c=c+1
2780        IF ben$(k)<=ben$(p) THEN 2830
2790         s=s+1
2800        SWAP ben$(k),ben$(p)
2810        SWAP cant(k),cant(p)
2815        SWAP pret(k),pret(p)
2820        f=1
2830      NEXT k
2840      IF f>0 THEN 2740
2850    WEND
2860    RETURN
2870    REM QUICK SORT(versiunea 1)
2880    n=iii:PRINT "Lista nesortata"
2890    i=n:GOSUB 2620
2900    c1=0:s1=0
2910     i1=1:j1=n
```

```
2920   i=il;j=jl:s=-1
2930   cl=cl+1
2940   IF ben$(i)<=ben$(j) THEN 2980
2950   sl=sl+1
2960   SWAP ben$(i),ben$(j)
2962   SWAP cant(i),cant(j)
2965   SWAP pret(i),pret(j)
2970   s=SGN(-s)
2980   IF s=1 THEN i=i+1 ELSE j=j-1
2990   IF i<j THEN 2930
3000   IF i+1>=jl THEN 3020
3010   p=p+1:s9(p,1)=i+1:s9(p,2)=jl
3020   jl=i-1
3030   IF il<jl THEN 2920
3040   IF p=0 THEN 3070
3050   il=s9(p,1):jl=s9(p,2):p=p-1
3060   GOTO 2920
3070   PRINT:PRINT "Lista sortata"
3080   I=N:GOSUB 2620
3090   PRINT:PRINT n;" beneficiari "
3100   PRINT cl;" comparatii"
3110   PRINT sl;" schimbari "
3120   PRINT
3130   RETURN
3200   REM QUICK SORT (varianta structurata)
3210   PRINT "Lista nesortata"
3215   i=iii:GOSUB 2620
3220   n=iii
3230   p=1:s9(1,1)=1:s9(1,2)=n
3240   WHILE p>0
3250     il=s9(p,1):jl=s9(p,2):p=p-1
3260     WHILE il<jl
3270       i=il:j=jl:s%=-1
3280       WHILE i<j
3290         IF ben$(i)<=ben$(j) THEN 3340
3300         SWAP ben$(i),ben$(j)
3310         SWAP cant(i),cant(j)
3320         SWAP pret(i),pret(j)
3330         s%=-(s%+1)
3340         IF NOT s% THEN j=j-1 ELSE i=i+1
3350       WEND
3360       IF i+1>=jl THEN 3400
3370       p=p+1
3380       s9(p,1)=i+1
3390       s9(p,2)=jl
3400       jl=i-1
3410     WEND
```

```
3420    WEND
3430    PRINT
3440    PRINT "Lista sortata"
3450    i=n:GOSUB 2620
3460    RETURN
```

BASIC-PLUS

```
100   REM PROGRAMUL SORTEAZA UN FISIER DE BENEFICIARI
110   DIM #1,B$(30)=40,C(30),P(30),B1$(30,30)
120   DIM S9(30,3),K$(30),N$(30),X(30),N1(30,2),K(30)
130   REM B$ - NUME BENEFICIAR
140   REM C - CANTITATE
150   REM P - PRET
160   REM S - SUMA
170   PRINT
180   GO SUB 200
190   GO TO 3400
200   REM ++++++++++++++++++++++++++++++++++++SORTARE DATE
210   GO SUB 3050 ! NUMARARE BENEFICIARI
220   GO SUB 2890 ! DESCHIDERE
230
240   GOSUB 510 ! CORP PROGRAM + MENIU
250   GO SUB 2950 ! INCHIDERE
260   RETURN
270   REM************************************** BUBBLE DEMO
280   N=I3
290   PRINT "         LISTA BENEFICIARI (NESORTATA)"
300   FOR I=1 TO N
310     PRINT B$(I),C(I),P(I)
320   NEXT I
330   PRINT:PRINT
340   FOR B=1 TO N-1
350     PRINT "PASUL --> ";B
360     F=0
370     FOR C=1 TO N-B
380       IF B$(C+1) >= B$(C) THEN 430
390       L$=B$(C):B$(C)=B$(C+1):B$(C+1)=L$
400       L=C(C):C(C)=C(C+1):C(C+1)=L
410       L1=P(C):P(C)=P(C+1):P(C+1)=L1
420       F=1
430     NEXT C
440     IF F=0 THEN 460
450   PRINT:NEXT B
460   PRINT:PRINT "       LISTA BENEFICIARI(SORTATA)"
470   FOR I=1 TO N
480     PRINT B$(I),C(I),P(I)
```

```
490     NEXT I
500     RETURN
510     PRINT:PRINT
520     PRINT
530     PRINT TAB(5);"METODA DE SORTARE:"
540     PRINT TAB(7);"1.BUBBLE DEMO"
550     PRINT TAB(7);"2.BUBBLE SORT"
560     PRINT TAB(7);"3.SORTARE PRIN EXTRACTIE"
570     PRINT TAB(7);"4.SORTARE PRIN INSERTIE"
580     PRINT TAB(7);"5.SORTARE INDEXATA"
590     PRINT TAB(7);"6.SHELL"
600     PRINT TAB(7);"7.QUIK SORT (VERSIUNEA 1)"
610     PRINT TAB(7);"8.QUICK SORT (VARIANTA STRUCTURATA)"
620     PRINT TAB(7);"9.SFIRSIT"
630     PRINT:PRINT TAB(5);:INPUT "OPTIUNEA DVS. ESTE ";M
640     IF M=1 THEN GO SUB 270
650     IF M=2 THEN GO SUB 800
660     IF M=3 THEN GO SUB 980
670     IF M=4 THEN GO SUB 1190
680     IF M=5 THEN GO SUB 1430
690     IF M=6 THEN GO SUB 1890
700     IF M=7 THEN GO SUB 2260
710     IF M=8 THEN GO SUB 2650
720     PRINT
730     RETURN
740
750     PRINT " LISTA NESORTATA : ":PRINT
760     PRINT "BENEFICIARI/CANTITATI LIVRATE/PRET":PRINT
770     FOR B=1 TO I3:PRINT B$(B);TAB(21);C(B);TAB(31);P(B)
780     NEXT B
790     RETURN
800     REM++++++++++++++++++++++++++++++++++++++SORTARE BUBBLE SORT
810     I=I3
820     REM
830     FOR M=1 TO I-1
840         F=0
850         FOR N=1 TO I-M
860             IF B$(N+1) >= B$(N) THEN 910
870             L$=B$(N):B$(N)=B$(N+1):B$(N+1)=L$
880             L=C(N):C(N)=C(N+1):C(N+1)=L
890             L1=P(N):P(N)=P(N+1):P(N+1)=L1
900             F=1
910         NEXT N
920         IF F=0 THEN 940
930     NEXT M
940     PRINT:PRINT "FISIER BENEFICIARI (SORTAT)"
950     FOR B=1 TO I:PRINT B$(B),C(B),P(B):NEXT B
```

```
960   PRINT
970   RETURN
980   REM++++++++++++++++++++++++++++++++++++++SORTARE PRIN EXTRACTIE
990   I=I3
1000  FOR M=1 TO I-1
1010  REM SE CONSIDERA CA B$(M)ESTE MAXIMUL
1020     M7$=B$(M)
1030     J=M
1040     REM
1050     REM
1060     FOR K=M+1 TO I
1070        IF B$(K) <=M7$ THEN 1100
1080        M7$=B$(K)
1090        J=K
1100     NEXT K
1110     REM
1120     L$=B$(M):B$(M)=B$(J):B$(J)=L$
1130     L=C(M):C(M)=C(J):C(J)=L
1140     L1=P(M);P(M)=P(J):P(J)=L1
1150   NEXT M
1160   PRINT "LISTA SORTATA :"
1170   FOR B=1 TO I:PRINT B$(B),C(B),P(B):NEXT B
1180   RETURN
1190   REM++++++++++++++++++++++++++++++++++++++SORTARE PRIN INSERTIE
1200   S5=1:D5=I3
1210   I=I3
1220   FOR M=S5+1 TO D5
1230     REM INSERTIE
1240     GOSUB 1300
1250   NEXT M
1260   PRINT "LISTA SORTATA :"
1270   FOR B=1 TO I:PRINT B$(B),C(B),P(B):NEXT B
1280   RETURN
1290   REM PROCEDURA DE INSERTIE
1300
1310   D=M-1
1320   U$=B$(M)
1330   C=C(M)
1340   P=P(M)
1350   IF D < S5 THEN 1400
1355   IF U$ >=B$(D) THEN 1400
1360   B$(D+1)=B$(D)
1370   C(D+1)=C(D):P(D+1)=P(D)
1380   D=D-1
1390   GOTO 1350
1400   C(D+1)=C:P(D+1)=P
1410   B$(D+1)=U$
```

```
1420   RETURN
1430   REM++++++++++++++++++++++++++++++++++++SORTARE INDEXATA
1440   N=I3
1450   FOR R=1 TO N
1460      N$(R)=B$(R)
1470      N1(R,1)=C(R)
1480      N1(R,2)=P(R)
1490   NEXT R
1500   INPUT "PENTRU CONTINUARE TASTATI (Y/N)";A$
1510   IF A$="N" THEN 1690
1520   INPUT "CIMPUL :1.BENEFICIAR/2.CANTITATE/3.PRET (1/2/3) ";J
1530   FOR R=1 TO N
1540      IF J=1 THEN 1570
1550      K(R)=N1(R,J-1)
1560      GO TO 1580
1570      K$(R)=N$(R)
1580   NEXT R
1590   GOSUB 1700
1600   PRINT "INREGISTRARI SORTATE DUPA CIMPUL ";J
1610   FOR R=1 TO N
1620      R1=X(R)
1630      PRINT N$(R1);" ",N1(R1,1);" ",N1(R1,2)
1640
1650   NEXT R
1660   PRINT
1670   INPUT "PENTRU CONTINUARE TASTATI (Y/N) ";A$
1680   IF A$="Y" THEN 1520
1690   RETURN
1700   IF J > 1 THEN 1800
1710   FOR A=1 TO N
1720      P=1
1730      FOR B=1 TO N
1740         IF K$(A) > K$(B) THEN P=P+1
1750         IF K$(A)=K$(B) AND A>B THEN P=P+1
1760      NEXT B
1770      X(P)=A
1780   NEXT A
1790   RETURN
1800   FOR A=1 TO N
1810      P=1
1820      FOR B=1 TO N
1830         IF K(A) > K(B) THEN P=P+1
1840         IF K(A)=K(B) AND A>B THEN P=P+1
1850      NEXT B
1860      X(P)=A
1870   NEXT A
1880   RETURN
```

```
1890  REM++++++++++++++++++++++++++SORTARE PRIN METODA SHELL
1900  I=I3:C=O:S=0
1910  PRINT "LISTA NESORTATA"
1920  GOSUB 1990
1930  REM SORTARE LISTA
1940  GOSUB 2070
1950  PRINT "LISTA SORTATA"
1960  GOSUB 1990
1970  PRINT C;"COMPARATII":PRINT S;"SCHIMBARI"
1980  RETURN
1990  REM RUTINA DE TIPARIRE PENTRU 5 COLOANE
2000  FOR K=1 TO I
2010    Z=K-5*INT((K-1)/5)
2020    PRINT TAB(10*(Z-1));B$(K);" ";
2030    IF Z >=5 THEN PRINT " "
2040  NEXT K
2050  PRINT " "
2060  RETURN
2070  REM METODA DE SORTARE SHELL
2080  G=I
2090  IF G <= 1 THEN 2250
2100  G=INT(G/2)
2110  M=I-G
2120  F=0
2130  FOR K=1 TO M
2140    P=K+G
2150    C=C+1
2160    IF B$(K) <= B$(P) THEN 2220
2170    S=S+1
2180    L$=B$(K):B$(K)=B$(P):B$(P)=L$
2190    L=C(K):C(K)=C(P):C(P)=L
2200    L1=P(K):P(K)=P(P):P(P)=L1
2210    F=1
2220  NEXT K
2230  IF F > 0 THEN 2120
2240  GO TO 2090
2250  RETURN
2260  REM++++++++++++++++++++++++++++++QUICK SORT (VERSIUNEA 1)
2270  N=I3:PRINT "LISTA NESORTATA"
2280  GOSUB 2580
2290  C1=0:S1=0
2300  I1=1:J1=N
2310  I=I1:J=J1:S=-1
2320  C1=C1+1
2330  IF B$(I) <= B$(J) THEN 2390
2340  S1=S1+1
2350  L$=B$(I):B$(I)=B$(J):B$(J)=L$
```

```
2360   L=C(I):C(I)=C(J):C(J)=L
2370   L1=P(I):P(I)=P(J):P(J)=L1
2380   S=SGN(-S)
2390   IF S=1 THEN 2420
2400   J=J-1
2410   GO TO 2430
2420   I=I+1
2430   IF I < J THEN 2320
2440   IF I+1 >=J1 THEN 2460
2450   P=P+1:S9(P,1)=I+1:S9(P,2)=J1
2460   J1=I-1
2470   IF I1 < J1 THEN 2310
2480   IF P=0 THEN 2510
2490   I1=S9(P,1):J1=S9(P,2):P=P-1
2500   GOTO 2310
2510   PRINT:PRINT "LISTA SORTATA"
2520   GOSUB 2580
2530   PRINT:PRINT N;" BENEFICIARI"
2540   PRINT C1;" COMPARATII"
2550   PRINT S1;" SCHIMBARI"
2560   PRINT
2570   GO TO 2640
2580   REM RUTINA DE TIPARIRE PENTRU 5 COLOANE
2590   FOR K=1 TO N
2600     Z=K-5*INT((K-1)/5)
2610     PRINT TAB(10*(Z-1));B$(K);" ";
2620     IF Z >=5 THEN PRINT " "
2630   NEXT K
2640   RETURN
2650   REM++++++++++++++++++++++++++++++QUICK SORT (VARIANTA STRUCTURATA)
2660   PRINT "LISTA NESORTATA"
2670   N=I3
2680   GO SUB 2580
2690   PRINT
2700   P=1
2710   S9(P,1)=1
2720   S9(P,2)=N
2730   IF P <= 0 THEN 2790
2740
2750   I1=S9(P,1)
2760   J1=S9(P,2)
2770   P=P-1
2780   GO SUB 3140
2790   GO TO 2730
2800   PRINT I3;" ARTICOLE"
2810   PRINT C1;" COMPARATII"
2820   PRINT C2;" SCHIMBARI"
```

```
2830    PRINT
2840    PRINT "LISTA SORTATA"
2850    GO SUB 2580
2860    PRINT
2870    RETURN
2880    REM++++++++++++++++++++++++++++++++++++++++ SUBRUTINE
2890    REM++++++++++++++++++++++++++++++++++++++++ DESCHIDERE FISIER
2900    OPEN "BENEF.DAT" AS VIRTUAL FILE 1
2910    PRINT
2920    PRINT "  ****   INCEPUT FAZA   ****  "
2930    PRINT
2940    RETURN
2950    REM+++++++++++++++++++++++++++++++++++++++INCHIDERE FISIER
2960    CLOSE 1
2970    PRINT
2980    PRINT "      ****    TERMINAT FAZA    ****      "
2990    PRINT
3000    RETURN
3010    REM+++++++++++++++++++++++++++++++++++++++++CAP TABEL
3020    PRINT "BENEFICIAR /CANTITATE LIVRATA /PRET"
3030    PRINT "====================================":PRINT
3040    RETURN
3050    REM++++++++++++++++++++NUMARARE BENEFICIARI INSCRISI IN FISIER
3060    OPEN "BENEF.DAT" AS VIRTUAL FILE 1
3070    I3=0
3080    FOR I=1 TO 30
3090      IF B$(I)="" THEN 3110
3100        I3=I3+1
3110    NEXT I
3120    CLOSE 1
3130    RETURN
3140    T=0 !************************************** SRT PT QUICK
3150    IF J1 <= I1 THEN 3390
3160    I=I1
3170    J=J1
3180    S%=-1
3190    IF I >=J THEN 3330
3200    C1=C1 + 1
3210    IF B$(I) <=B$(J) THEN 3270
3220    L$=B$(I):B$(I)=B$(J):B$(J)=L$
3230    L=C(I):C(I)=C(J):C(J)=L
3240    L1=P(I):P(I)=P(J):P(J)=L1
3250    C2=C2 + 1
3260    S%=-S%
3270    IF S% < 0 THEN 3300
3280    J=J - 1
3290    GO TO 3190
```

```
3300    I=I + 1
3310    GO TO 3190
3320
3330    IF I+1 >= J1 THEN 3370
3340    P=P+1
3350    S9(P,1)=I+1
3360    S9(P,2)=J1
3370    J1=I-1
3380    GO TO 3150
3390    RETURN
3400    STOP
3410    END
```

EXEMPLELE 12

BASIC-AMSTRAD (I)

```
5       REM "TRASARE REZERVOR - DISPLAY CU CARACTERE MOZAIC"
10      CLS
140     z=8
145     i=0
150     LOCATE z,20:PRINT CHR$(130);CHR$(131);CHR$(129)
155     FOR n=1 TO 11
160        LOCATE z-1,20-n:PRINT"-" CHR$(138);CHR$(128);CHR$(133)
165     NEXT n
170     LOCATE z,8:PRINT CHR$(136);CHR$(140);CHR$(132)
175     FOR n=1 TO 11
180        LOCATE (z-6),20-n:PRINT i
185        i=i+10
190     NEXT n
192     FOR j=z+1 TO 38
194        LOCATE j,20:PRINT CHR$(131)
195     NEXT j
196     LOCATE 15,22:PRINT"L M M J V S D "
205     LOCATE 1,1:PRINT"Cantitatea de combustibil rezervor ";d;
206     INPUT t
207     LOCATE 1,1:PRINT"
208     GOSUB 500
210     FOR z=14 TO 32 STEP3
212        READ t
213        DATA 10,20,0,15,30,20,5
214        GOSUB 500
215     NEXT z
350     GOTO 350
```

```
500    REM "Subrutina de vizualizare a cant. de comb."
510    FOR n=1 TO 11
520      LOCATE z+1,20-n:PRINT CHR$(128)
525    NEXT n
530    y=INT( (t+5)/5+0.5)
535    FOR n=1 TO INT(y/2)
540      LOCATE z+1,20-n:PRINT CHR$(143)
545    NEXT n
550    IF INT(y/2)=y/2 THEN 650
560    y=INT(y/2)
565    LOCATE z+1,19-y:PRINT CHR$(140)
650    RETURN
```

BASIC-AMSTRAD(II)

```
5     REM Generarea a trei rezervoare display-Metoda punct cu punct
10    READ y
20    DATA 50
30    CLS
40    FOR x=100 TO 630 STEP200
50        MOVE x,y
60        DRAWR 70,0
70        DRAWR 0,300
80        DRAWR -70,0
90        DRAWR 0,-300
100       q=50
110       k=0
120       GOSUB 340
130       FOR i=y+50 TO y+250 STEP8
140           MOVE x,i
150           r=(i-y-50)/40
160           IF INT (r)<>r THEN 220
170           DRAWR -20,0
180           LOCATE (x-70)/16,(400-i)/16
190           PRINT k;
200           k=k+10
210           GOTO 230
220           DRAWR -10,0
230       NEXT i
240   NEXT x
250   q1=30
260   FOR x=100 TO 630 STEP200
262       d=d+1
265       LOCATE 1,1:PRINT"CANTITATEA IN TONE REZERVOR "D;
280       INPUT t
290       p=50
300       q=t*4
```

```
310        GOSUB 340
315        LOCATE 1,1:PRINT"
320     NEXT x
325     GOTO 325
330     STOP
340     MOVE x,y+p
350     FOR i=y+p TO y+p-1+q
360        FOR j=x-1+ql TO x+70-ql
370           DRAWR 0,1
380           MOVE j+1,i
390        NEXT j
400     NEXT i
410     DRAWR 0,1
420     RETURN
```

BASIC-AMSTRAD(III)

```
5       REM Trasarea histogramelor. Metoda "linie cu linie"
10      READ x,y
20      DATA 80,100
30      FOR d=1 TO 3
40         CLS
50         k=0
60         x=80
70         MOVE x,350
80         DRAW x,100:DRAW 640,100
90         MOVE x-10,340
100        DRAW x,350
110        DRAW x+10,340
114        MOVE 630,y+10
115        DRAW 640,y
116        DRAW 630,y-10
120        FOR i=y TO y+200 STEP8
130           MOVE x,i
140           r=(i-y)/40
150           IF INT(r)<>r THEN 210
160           DRAWR -20,0
170           LOCATE(x-70)/16,(400-i)/16
180           PRINT k
190           k=k+10
200           GOTO 220
210           DRAWR -10,0
220        NEXT i
230        ql=10
240        LOCATE 1,1:PRINT"LIVRARI BENZINA REZERVOR"D"PE ZILE"
250        LOCATE(x+16)/16,22:PRINT "Luni Marti Mierc. Joi Vineri Simb."
```

```
260         FOR x=80 TO 620 STEP95
270            READ t
280            DATA 5,15,10,7,9,40
290            DATA 4,5,12,7,8,9,23
300            DATA 1,2,3,4,5,6,7,8
310            q=t*4
320            GOSUB 360
330         NEXT x
335         IF INKEY$="" GOTO 335
340      NEXT d
350      GOTO 350
360      MOVE x,y+p
370      FOR i=y+p TO y+p-l+q
380         MOVE x-l+ql,i
390         DRAW x+70-ql,i
420      NEXT i
440      RETURN
```

BASIC-AMSTRAD(IV)

```
5       REM Trasarea histogramelor in 3D
10      READ x,y
20      DATA 80,100
30      FOR d=1 TO 3
40         CLS
50         k=0
60         x=80
70         MOVE x,350
80         DRAW x,100:DRAW 640,100
90         MOVE x-10,340
100        DRAW x,350
110        DRAW x+10,340
114        MOVE 630,y+10
115        DRAW  640,y
116        DRAW 630,y-10
120        FOR i=y TO y+200 STEP8
130           MOVE x,i
140           r=(i-y)/40
150           IF INT(r)<>r THEN 210
160           DRAWR -20,0
170           LOCATE(x-70)/16,(400-i)/16
180           PRINT k
190           k=k+10
200           GOTO 220
210           DRAWR -10,0
220        NEXT i
```

```
230       ql=10
240   LOCATE 1,1:PRINT"LIVRARI BENZINA REZERVOR"D"PE ZILE"
250   LOCATE(x+16)/16,22:PRINT "Luni Marti Mierc. Joi Vineri Simb."
260   FOR x=80 TO 620 STEP95
270      READ t
280      DATA 5,15,10,7,9,40
290      DATA 4,5,12,7,8,9,23
300      DATA 1,2,3,4,5,6,7,8
310      q=t*4
320      GOSUB 360
325      DRAW x+ql+9,i+9
326      DRAW x-1+ql,i-1
330   NEXT X
335   IF INKEY$ ="" GOTO 335
340  NEXT d
350  GOTO 350
360  MOVE x,y+p
370  FOR i=y+p TO y+p-1+q
380   MOVE x-1+ql,i
390   DRAW x+70-ql,i
395   DRAW x+70 -ql+10,i+10
420  NEXT i
440  RETURN
```

BASIC HC-85,TIM S,SPECTRUM

```
  5 REM Punct cu punct
 10 READ y
 20 DATA 20
 30 CLS
 35 LET p=0:LET ql=0
 40 FOR x=40 TO 200 STEP80
 50   PLOT x,y
 60   DRAW 30,0
 70   DRAW 0,130
 80   DRAW -30,0
 90   DRAW 0,-130
100   LET q=20
110   LET k=0
115   LET ql=28
120   GO SUB 340
130   FOR i=y+20 TO y+120 STEP
4
140     PLOT x,I
150     LET r=(i-y-20)/20
160      IF INT (r) <>r THEN GO
TO 220
170     DRAW -20,0
180     PRINT AT (175-i)/8,(x-
30)/8;k
200     LET k=k+10
210     GO TO 230
220     DRAW -10,0
230   NEXT i
240 NEXT x
250 LET ql=2
255 LET d=0
260 FOR x=40 TO 200 STEP80
262   LET d=d+1
265   PRINT AT 1,1;"cantitatea
in tore rezervor";d;
280   INPUT t
290   LET p=20
300   LET q=t*2
310   GO SUB 340
315   PRINT AT 1,1;"
                        "
```

```
320 NEXT x
325 GO TO 325
330 STOP
340 PLOT x,y+p
350 FOR i=y+p TO y+p-1+q
360   FOR j=x+(30-ql)/2 TO x+(3
0-ql)/2+ql
370     DRAW 0,1
380     PLOT j,i
390   NEXT j
400 NEXT i
410 DRAW 0,1
420 RETURN
```

BASIC HC-85,TIM S,SPECTRUM(II)

```
  5 REM Linie cu linie
 10 READ x,y
 20 DATA 40,20
 25 FOR d=1 TO 3
 27   LET x=40
 30   CLS
 32   LET k=0
 35   LET p=0:LET ql=0
 50   PLOT x,y
 60   DRAW 200,0
 70   DRAW -5,5
 80   PLOT x+200,y
 90   DRAW -5,-5
 91   PLOT x,y
 92   DRAW 0,137
 93   DRAW -5,-5
 94   PLOT x,137+y
 95   DRAW 5,-5
100   LET q=20
110   LET k=0
115   LET ql=28
130   FOR i=y+20 TO y+120 STEP
4
140     PLOT x,i
150     LET r=(i-y)/20
160     IF INT (r) <>r THEN GO
TO 220
170     DRAW -20,0
180     PRINT AT (175-i)/8,(x-3
0)/8;k
200       LET k=k+10
210       GO TO 230
220       DRAW -10,0
230   NEXT i
250   LET ql=8
265   PRINT AT 1,1;"Livrari ben
zina rezervor";d;
270   PRINT AT (175-y)/8+1,x/8;
"L  M  M  J  V  S"
275   FOR x=40 TO 190 STEP30
280     READ t
285     DATA 5,15,10,7,9,40
286     DATA 4,5,12,7,8,9,23
287     DATA 1,2,3,4,5,6,7,8
290     LET p=20
300     LET q=t*2
310     GO SUB 350
320   NEXT x
321   IF INKEY$=""THEN GO TO 32
1
322 NEXT d
325 GO TO 325
350 FOR i=y+p TO y+p-1+q
360   PLOT x+(30-ql)/2,i
370   DRAW ql,0
400 NEXT i
420 RETURN
```

BASIC HC-85,TIM S,SPECTRUM(III)

```
  5 REM Histograme 3D
 10 READ x,y
 20 DATA 40,20
 25 FOR d=1 TO 3
 27   LET x=40
 30   CLS
 32   LET k=0
 35   LET p=0: LET ql=0
 50   PLOT x,y
 60   DRAW 200,0
 70   DRAW -5,5
 80   PLOT x+200,y
 90   DRAW -5,-5
 91   PLOT x,y
 92   DRAW 0,137
```

```
 93   DRAW -5,-5
 94   PLOT x,137+y
 95   DRAW 5,-5
 100  LET q=20
 110  LET k=0
 115  LET ql=28
 130  FOR i=y+20 TO y+120 STEP
4
 140     PLOT x,i
 150     LET r=(i-y)/20
 160     IF INT (r)<>r THEN GO T
O 220
 170     DRAW -20,0
 180     PRINT AT (175-i)/8,(x-3
0)/8;k
 200     LET k=k+10
 210     GO TO 230
 220     DRAW -10,0
 230  NEXT i
 250  LET ql=8
 265  PRINT AT 1,1;"Livrari ben
zina rezervor";d;
 270  PRINT AT (175-y)/8+1,x/8;
"L  M  M  J  V  S"
```

```
 275  FOR x=40 TO 190 STEP30
 280     READ t
 285     DATA 5,15,10,7,9,40
 286     DATA 4,5,12,7,8,9,23
 287     DATA 1,2,3,4,5,6,7,8
 290     LET p=20
 300     LET q=t*2
 310     GO SUB 350
 320  NEXT x
 321  IF INKEY$="" THEN GO TO 3
21
 322 NEXT d
 325 GO TO 325
 330 STOP
 350 FOR i=y+p TO y+p-1+q
 360   PLOT x+(30-ql)/2,i
 370   DRAW ql,0
 375   DRAW 5,5
 400 NEXT i
 405 DRAW -Q1,0
 410 DRAW -5,-5
 420 RETURN
```

BASIC-80

```
10    DIM A(6,3),B(6),ZIL$(6)
20    DATA LUNI,MARTI,MIERCURI,JOI,VINERI,SIMBATA
30    FOR I=1 TO 6:READ ZIL$(I):NEXT I
40    FOR MM=1 TO 25:PRINT:NEXT MM
50    PRINT "**************************************************"
60    PRINT "*  Program de reprezentare grafica a situatiei  *"
70    PRINT "*    livrarilor de benzina din trei rezervoare   *"
80    PRINT "**************************************************"
90    PRINT:PRINT:PRINT
100   MAX=0:MAXB=0
110   FOR I=1 TO 6
120      FOR J=1 TO 3
130         PRINT "*";ZIL$(I);TAB(13);"R";:PRINT USING "#";J:PRINT " ...
140         INPUT A(I,J)
150         IF(MAX<A(I,J))THEN MAX=A(I,J)
160      NEXT J
170      B(I)=(A(I,1)+A(I,2)+A(I,3))/3
180      IF(MAXB<B(I)) THEN MAXB=B(I)
190   NEXT I
```

```
200     PRINT CHR$(27)+"I"+CHR$(27)+"2"
210     VIEWPORT 0,132,0,99:WINDOW 0,1320,0,1000
220     MOVE 0,0:DRAW 0,1000:DRAW 1320,1000:DRAW 1320,0:DRAW 0,0
230     XO=120
240     FOR K=1 TO 6
250         X=X0+(K-1)*200
260         GOSUB 340
270     NEXT K
280     GOSUB 840
290     GOSUB 950
300     GOSUB 1230
310     U$=INPUT$(1)
320     GOSUB 1370
330     END
335     REM HISTOGRAME 3D
340     Y=100
350     Y1=INT(A(K,1)*800/MAX)
360     Y2=INT(A(K,2)*800/MAX)
370     Y3=INT(A(K,3)*800/MAX)
380     MOVE X,Y
390     RDRAW 0,Y1:RDRAW 40,0:RDRAW 0,-Y1:RDRAW -40,0
400     YY1=Y
410     WHILE YY1<Y1+91
420         YY1=YY1+10
430         MOVE X,YY1:RDRAW 40,0
440     WEND
450     IX=X;IY1=Y1:IY2=Y2:GOSUB 680
460     MOVE 40+X,Y
470     RDRAW 0,Y2:RDRAW 40,0:RDRAW 0,-Y2:RDRAW -40,0
480     X1=X+40
490     WHILE X1<X+80
500         X1=X1+2
510     MOVE X1,Y:RDRAW 0,Y2
520     WEND
530     IX=X+40:IY1=Y2:IY2=Y3:GOSUB 680
540     MOVE 80+X,Y
550     RDRAW 0,Y3:RDRAW 40,0:RDRAW 0,-Y3:RDRAW -40,0
560     X1=X+80
570     FOR MM=1 TO 3
580         X1=X1+10
590         MOVE X1,Y:RDRAW 0,Y3
600     NEXT MM
610     YY1=Y+15
620     WHILE YY1<Y3+100
630         MOVE X+80,YY1:RDRAW 40,0
640         YY1=YY1+15
650     WEND
```

```
660     GOSUB 800
670     RETURN
675     REM Tratarea cazului "linii ascunse in 3D"
680     IF(IY1>=IY2-20)THEN 710
690     MOVE IX,Y+IY1:RDRAW 30,20:RDRAW 10,0
700     GOTO 790
710     IF(IY1<IY2-20)OR(IY1>IY2)THEN 750
720     YY1=INT(37-(IY2-IY1))
730     MOVE IX,Y+IY1:RDRAW 30,20:RDRAW YY1,0
740     GOTO 790
750     REM
760     YY1=IY1-IY2
770     MOVE IX,Y+IY1:RDRAW 30,20:RDRAW 40,0:RDRAW -30,-20
780     RMOVE 30,20:RDRAW 0,-YY1
790     RETURN
800     MOVE X+80,Y3+100
810     RDRAW 30,20:RDRAW 40,0:RDRAW -30,-20
820     MOVE X+120,Y:RDRAW 30,20:RDRAW 0,Y3
830     RETURN
835     REM Comentarii sub histograma in 3D
840     PRINT CHR$(27)+"l"+CHR$(45)+CHR$(33);
850     PRINT "*** Situatia livrarilor zilnice la trei rezervoare ***"
860     YY1=41:GOSUB 930:PRINT "Luni"
870     YY1=53:GOSUB 930:PRINT "Marti"
880     YY1=63:GOSUB 930:PRINT "Miercuri"
890     YY1=78:GOSUB 930:PRINT "Joi"
900     YY1=88:GOSUB 930:PRINT "Vineri"
910     YY1=100:GOSUB 930:PRINT "Simbata"
920     RETURN
930     PRINT CHR$(27)+"l"+CHR$(YY1)+CHR$(54);
940     RETURN
945     REM Legenda histograma 3D
950     MOVE 40,850:RDRAW 0,40:RDRAW 40,0:RDRAW 0,-40:RDRAW -40,0
960     MOVE 40,890:RDRAW 30,20:RDRAW 40,0:RDRAW -30,-20
970     YY1=10
980     XX1=850+YY1
990     RMOVE 30,20:RDRAW 0,-40:RDRAW -30,-20
1000    WHILE XX1<889
1010       MOVE 40,XX1:RDRAW 40,0
1020       XX1=XX1+YY1
1030    WEND
1040    MOVE 40,550:RDRAW 0,40:RDRAW 40,0:RDRAW 0,-40:RDRAW -40,0
1050    MOVE 40,590:RDRAW 30,20:RDRAW 40,0:RDRAW -30,-20
1060    RMOVE 30,20:RDRAW 0,-40:RDRAW -30,-20
1070    YY1=2
1080    XX1=42
1090    WHILE XX1<79
```

```
1100      MOVE XX1,550:RDRAW 0,40
1110      XX1=XX1+2
1120    WEND
1130    MOVE 40,250:RDRAW 0,40:RDRAW 40,0:RDRAW 0,-40:RDRAW -40,0
1140    MOVE 40,290:RDRAW 30,20:RDRAW 40,0:RDRAW -30,-20
1150    RMOVE 30,20:RDRAW 0,-40:RDRAW -30,-20
1160    FOR MM=1 TO 3:MOVE 40+MM*10,250:RDRAW 0,40:NEXT MM
1170    YY1=265
1180    WHILE YY1<285
1190      MOVE 40,YY1:RDRAW 40,0
1200      YY1=YY1+15
1210    WEND
1220    RETURN
1230    YY1=37:GOSUB 1270:PRINT "Rez.1"
1240    YY1=44:GOSUB 1270:PRINT "Rez.2"
1250    YY1=51:GOSUB 1270:PRINT "Rez.3"
1260    RETURN
1270    PRINT CHR$(27)+"1"+CHR$(33)+CHR$(YY1);
1280    RETURN
1285    REM Histograme 2D
1290    POKE 5268,81
1300    POKE 52A7,41
1310    POKE 52B4,81
1320    RETURN
1330    POKE 5268,88
1340    POKE 52A7,48
1350    POKE 52B4,88
1360    RETURN
1370    PRINT CHR$(27)+"I"+CHR$(27)+"2"
1380    VIEWPORT 0,133,0,99
1390    WINDOW 0,1320,0,1000
1400    GOSUB 1290
1410    GCLEAR
1420    GOSUB 1330
1430    Y=10:X=13
1440    FOR MM=1 TO 6
1450      VIEWPORT X,X+16,Y,Y+80
1460      GCLEAR
1470      X=13+MM*20
1480    NEXT MM
1490    YY1=15:X1=10:X2=125
1500    FOR MM=1 TO 8
1510      VIEWPORT X1,X2,YY1,YY1+1
1520      GCLEAR
1530      YY1=YY1+10
1540    NEXT MM
1550    GOSUB 1290
```

```
1560   FOR MM=1 TO 6
1570      X1=15+(MM-1)*20
1580      X2=X1+12
1590      Y1=15:Y2=INT(70*B(MM)/MAXB)+15
1600      VIEWPORT X1,X2,Y1,Y2
1610      GCLEAR
1620   NEXT MM
1630   PRINT CHR$(27)+"1"+CHR$(52)+CHR$(33);
1640   PRINT " ** Situatia livrarilor medii zilnice **"
1650   PRINT CHR$(14):REM Scriere in video invers
1660   YY1=43:GOSUB 930:PRINT "Luni"
1670   YY1=55:GOSUB 930:PRINT "Marti"
1680   YY1=65:GOSUB 930:PRINT "Miercuri"
1690   YY1=80:GOSUB 930:PRINT "Joi"
1700   YY1=90:GOSUB 930:PRINT "Vineri"
1710   YY1=101:GOSUB 930:PRINT "Simbata"
1720   PRINT CHR$(15)
1730   PRINT CHR$(27)+"1"+CHR$(33)+CHR$(35)
1740   PRINT USING "##.#";MAXB
1750   PRINT CHR$(27)+"1"+CHR$(34)+CHR$(52);" 0 "
1760   U$=INPUT$(1)
1770   PRINT CHR$(27)+"2"
1780   RETURN
```

EXEMPLUL 13

BASIC-AMSTRAD

```
1    REM  Animatie la nivel de caracter
2    CLS
3    d=10
4    n=3
5    ll=18
6    kl=2
7    p=5
8    DIM t(p),r(p)
9    DIM x(p+1),v(p,3),c(n,p),z(p)
10   v(3,3)=6
11   v(2,3)=2
12   v(1,3)=6
13   v(1,2)=1
14   v(2,2)=1
```

```
15    v(3,2)=1
20    FOR i=1 TO n
22       FOR j=1 TO 3
23          READ c(i,j)
24       NEXT j
25    NEXT i
27    DATA 10,20,5,10,20,0,17,0,0
30    CLS
50    READ x(1),y,h,hl,l
55    DATA 5,8,6,5,3
60    FOR j=1 TO p-2
65       x(j+1)=x(j)+d
70       FOR i=y TO y+h
75          LOCATE x(j)-l,i:PRINT CHR$(143)
80          LOCATE x(j)+l,i:PRINT CHR$(143)
85       NEXT i
110      FOR j1=x(j)-l+1 TO x(j)+l
115         LOCATE j1,y+h:PRINT CHR$(143)
120      NEXT j1
130      FOR j2=y-1 TO y-l+1 STEP-1
135         i1=i1+1
140         LOCATE x(j)-l+i1,j2:PRINT CHR$(143)
145         LOCATE x(j)+l-i1,j2:PRINT CHR$(143)
150      NEXT j2
151      LOCATE x(j),y-l+1:PRINT CHR$(143)+CHR$(143)
152      i1=0
153      GOSUB 1100
157      GOSUB 950
160   NEXT j
161   FOR j=1 TO 40
162      LOCATE j,l1+i1+5:PRINT CHR$(143)
163   NEXT j
165   i1=3
167   j1=6
200   b=1
203   LOCATE 13,25:PRINT"BENEFICIARUL";b
205   FOR j=3 TO 1 STEP-1
206      t(j)=0
207      z(j)=x(j)
210      IF c(b,j)=0 THEN 235
220      GOSUB 350
230      GOSUB 400
231      LOCATE x(j)-3,1:PRINT"cererea"
232      LOCATE x(j)-1,2:PRINT c(b,j)
235   NEXT j
240   FOR k=1 TO p-2
245      IF c(b,k)=0 THEN 275
```

```
250     IF v(k,1)<v(k,2) THEN 265
255     GOSUB 750
260     GOTO 275
265     j=k
266     GOSUB 950
270     GOSUB 750
275   NEXT k
280   FOR k=1 TO p-2
285     IF c(b,k)<>0 THEN 240
286     GOSUB 1000
290   NEXT k
295   b=b+1
300   IF b>n THEN 1500
305   GOTO 203
340   REM"Generare vehicul"
350   FOR k=0 TO j1
355     FOR i=0 TO i1
360       LOCATE k1+k,l1+i:PRINT CHR$(143)
365     NEXT i
370     LOCATE k1+k,l1+i1+1:PRINT "O"
375   NEXT k
380   RETURN
390   REM"Deplasare vehicul"
400   FOR k=k1 TO z(j)
405     FOR i=0 TO i1
410       LOCATE k+j1+1,l1+i:PRINT CHR$(143)
415       LOCATE k,l1+i:PRINT" "
420     NEXT i
425     LOCATE k+j1+1,l1+i1+1:PRINT"O"
430     LOCATE k,l1+i1+1:PRINT " "
435   NEXT k
440   RETURN
750   yl=y+h-v(k,1)
755   FOR j3=x(k)-l+1 TO x(k)+l-1
760     LOCATE j3,yl:PRINT" "
765   NEXT j3
780   v(k,1)=v(k,1)-1
782   c(b,k)=c(b,k)-1
785   LOCATE x(k)-1,2:PRINT c(b,k)
786   t(k)=t(k)+1
787   FOR a=yl+1 TO  l1-1
788     LOCATE z(k)+4,a:PRINT CHR$(143)
789   NEXT a
790   LOCATE z(k)+3,l1+2:PRINT t(k)
791   FOR a=yl+1 TO l1-1
792     LOCATE z(k)+4,a:PRINT" "
793   NEXT a
```

```
795  RETURN
950  FOR yl=y+h-v(1,2) TO y+h-v(1,3) STEP-1
955    FOR j3=x(j)-1+1
960      LOCATE j3,yl:PRINT CHR$(143)
965    NEXT j3
966    v(j,1)=v(j,1)+1
970  NEXT yl
975  v(j,1)=v(j,3)
980  RETURN
1000 FOR i=0 TO il+1
1005   FOR j=0 TO jl+1
1010     LOCATE z(k)+j,ll+i:PRINT " "
1015   NEXT j
1020 NEXT i
1025 RETURN
1100 IF j=1 THEN 1120
1105 IF j=2 THEN 1130
1110 LOCATE x(j)-3,4:PRINT"<CO 75>"
1115 GOTO 1135
1120 LOCATE x(j)-3,4:PRINT"<CO 98>"
1125 GOTO 1135
1130 LOCATE x(j)-3,4:PRINT"<CO 90>"
1135 RETURN
1500 GOTO 1500
```

TEMA 14

☐ BASIC HC-85, TIM S, SPECTRUM

```
  15 CLS : FOR i=0 TO 255 STEP 4
  20 PLOT 0,87: DRAW i,87: BEEP
.01,i/6
  25 NEXT i
  30 FOR i=0 TO 255 STEP 4
  35 PLOT 255,88: DRAW -i,-87: B
EEP .01,i/6
  40 NEXT i
  45 FOR i=255 TO 0 STEP -4
  50 PLOT 255,175: DRAW -i,-88:
BEEP .01,i/6
  55 NEXT i
  60 FOR i=0 TO 255 STEP 4
  65 PLOT 0,0: DRAW i,88: BEEP .
01,i/6
  70 NEXT i
  75 FOR i=-40 TO 40 STEP 4: BEE
P .01,i: NEXT i
  80 PRINT AT 19,23;"SIMULATOR";
AT 20,20;"Baza livrare";AT 21,14
;"produse petroliere": BEEP .1,4
0

  85 PRINT AT 0,0;"D.I."
  91 LOAD "UDG"CODE
  92 CLEAR 49999: LOAD "r/w. mem
"CODE
  93 LOAD "ParcRD"SCREEN$ : RAND
OMIZE USR 50021
  94 MERGE "": GO TO 10
  95 REM SAVE "start" LINE 1
 999 STOP
1000 REM
          --- Initializari ---

1010 DIM x$(3,7,10)
1020 DIM a$(16): DIM b$(16): DIM
 c$(16): DIM d$(16)
1030 LET a$="                ":
LET b$="/              ": LET c
$="                ":
1045 LET d$="          wW  ":
LET e$="              ": LET f
$="                "
1060 DIM w(3): LET w(1)=112: LET

 w(2)=56: LET w(3)=0
1070 DIM c(6): FOR i=1 TO 6: REA
D c(i): NEXT i
1075 DATA 22,20,16,12,9,4.5
1080 DIM h$(19,2): FOR k=1 TO 19
: READ h$(k): NEXT k
1085 DATA "AG","B ","BC","BT","B
R","BZ","CJ","CT","DB","DJ","GL"
,"IS","IF","MH","MS","NT","PH","
SU","UN"
```

```
1100 DIM g$(3,12,10): DIM v(3,12
)
1115 DIM i(3): DIM y(3): DIM z(3
): DIM x(3): DIM u(3): DIM r(3)
1120 DIM t(3)
1125 LET t(1)=0: LET t(2)=0: LET
 t(3)=0
1128 DIM a(12,3): DIM i$(11,5)
1130 FOR b=1 TO 11: FOR c=1 TO 3
1135 READ a(b,c)
1137 NEXT c
1138 READ i$(b)
1140 NEXT b
1145 DATA 12,40,20,"Ian. ",10,45
,23,"Feb. ",11,38,21,"Mart.",9,4
2,19,"Apr. ",7,50,21,"Mai  ",6,5
5,19,"Iun. ",9,46,23,"Iul. ",10,
46,23,"Aug. ",14,35,18,"Sept.",1
2,43,19,"Oct. ",5,21,9,"Nov. "
1150 DIM f(3): DIM g(3): LET f(1
)=1: LET f(2)=5: LET f(3)=3
1155 RESTORE 1160: DIM H(36): FO
R k=1 TO 30: READ H(k): NEXT k
1160 DATA 61,38,15,57,35,12,64,3
8,18,55,32,16,65,36,18,57,28,16,
70,45,11,69,38,13,62,46,16,55,20
,15
1500 REM
         --- Init. sistem ---

1502 RANDOMIZE USR 50000
1505 BORDER 6
1510 LET sum=0
1515 LET o=7: LET m=0: LET pre=3


1550 FOR n=1 TO 3
1553 GO SUB 8600
1555 GO SUB 8517
1557 BEEP .1,6
1560 NEXT n
1570 LET d=1
1575 LET x(1)=35: LET x(2)=35: L
ET x(3)=35
1580 FOR n=1 TO 3
1585 BEEP .04,40: GO SUB 8435: G
O SUB 8210
1590 NEXT n
1595 LET secv=0
2100 REM
        *** MASTER PROG. ***
          **************
2105 REM
              - ceas -

2110 PRINT AT 0,27;o;AT 0,29;":"
;m
2120 IF o=14 THEN PRINT AT 0,0;


FLASH 1;"Sfirsitul prog. de lucr
u": IF m<=5 THEN GO SUB 8300
2130 IF m>=60 THEN LET m=0: LET
o=o+1: PRINT AT 0,31;" ": PRINT
AT 0,27;o
2140 LET m=m+2: PRINT AT 0,30;m
```

```
2150 BEEP .1,45: PAUSE 1: BEEP .
15,45
2299 REM
          - rezervoare -
2300 FOR n=1 TO 3
2310 PLOT OVER 1;23,w(n)+3+x(n)
2315 DRAW OVER 1;70,0
2317 LET g(n)=g(n)+1
2320 IF g(n)>=f(n) THEN LET g(n)
=0: LET x(n)=x(n)-1
2324 IF x(n)<=0 THEN GO SUB 8200
: GO SUB 8600: GO SUB 8210
2327 REM
          - cisterne -
2330 LET i(n)=i(n)+.25
2335 IF i(n)<y(n) THEN GO TO 234
7
2337 LET y(n)=y(n)+r(n)
2340 IF INT (y(n)/(r(n)+.0001))>
=8 THEN LET inc=136: LET lung=80
: GO TO 2345
2343 LET inc=168: LET lung=16
2345 PLOT OVER 1;inc,w(n)+8+INT
(y(n)/(r(n)+.0001)): DRAW OVER 1
;lung,0
2347 IF y(n)/(r(n)+.0001)>16 THE
N GO SUB 8300: PRINT FLASH 1;AT
n*7-4,19;"■": GO SUB 8200: GO SU
B 8400: GO SUB 8517: BEEP .04,32
: PAUSE 1: BEEP .08,32: GO SUB 8
435: GO SUB 8210: PRINT FLASH 0;
AT n*7-4,19;" "
2353 REM
          - incarcat-timp-max. -
2355 IF d=3 THEN PRINT BRIGHT 1;
AT 7*n-5,23;"         ";AT 7*n-5,
23;"c ";i(n);AT 7*n-5,30;"t."
2357 IF d=6 THEN PRINT AT 7*n-5,
23;"         ";AT 7*n-5,23;"max "
;v(n,t(n));AT 7*n-5,30;"t."
2360 IF d=9 THEN PRINT PAPER 6;A
T 7*n-5,23;"         ";AT 7*n-5,2
3;"t ";INT ((v(n,t(n))-i(n))*8);
AT 7*n-5,28;" min"
2365 NEXT n
2370 LET d=d+1: IF d>9 THEN LET
d=1
2380 REM
          - interogare tastatura -
2385 LET l$=INKEY$
2390 IF l$="a" THEN RANDOMIZE US
R 50021: GO SUB 7500: RANDOMIZE
USR 50000
2395 IF l$="o" THEN RANDOMIZE US
R 50021: GO SUB 7700: RANDOMIZE
USR 50000

2400 IF l$="h" THEN RANDOMIZE US
R 50021: GO SUB 7000: RANDOMIZE
USR 50000
```

```
2405 IF secv=1 THEN PAUSE 0
2410 GO TO 2100
7000 REM
       --- Histograma 3 dim ---

7001 PRINT AT 0,0;"   Auto   Opt
iuni   Hist    "
7002 PRINT OVER 1;AT 0,19;"██████
██"; OVER 0; PAPER 5;AT 1,10;"
               ";AT 2,10;"
 Histograma tri-";AT 3,10;" -
dimensionala :  ";AT 4,10;"plan
oriz.:-produse ";AT 5,10;"
      -luni   ";AT 6,10;"vertica
la:-planific.";AT 7,10;"
   -realizat";AT 8,10;"
       "
7004 PRINT PAPER 5;AT 9,10;"▀▣ R
evenire la sch. ";AT 10,10;"


                 "
7010 PAUSE 0: BEEP .15,30: PAUSE
 0: BEEP .1,40
7018 FOR k=2 TO 19
7020 PRINT PAPER 0;AT k,1;"
                 "
7025 NEXT k
7030 PRINT INK 7; PAPER 0;AT 16,
1;"Motor-";AT 17,1;" Petrol-";AT
 18,1;"  Benzina-"
7035 PRINT INK 7; PAPER 0;AT 19,
11;"DN";AT 18,13;"OS";AT 17,15;"
AI";AT 16,17;"IM";AT 15,19;"AM";
AT 14,21;"FI"
7040 PLOT 47,49: DRAW INK 7;0,80
7045 FOR n=48 TO 128 STEP 8: PLO
T INK 7;46,n: NEXT n
7050 FOR n=48 TO 128 STEP 16: PL
OT INK 7;44,n: NEXT n
7055 PRINT INK 7; PAPER 0;AT 3,2
;"mii";AT 4,2;"  tone";AT 5,2;"1
50";AT 7,2;"120";AT 9,2;" 90";AT


 11,2;" 60";AT 13,2;" 30";AT 15,
2;"  0"
7060 BEEP .05,23
7065 LET j=0
7070 LET cx=88: LET cy=28
7075 LET dx=6: LET dy=3
7080 GO SUB 7140
7085 FOR x=88 TO 0 STEP -8
7090 FOR y=40 TO 0 STEP -20
7095 LET j=j+1
7097 LET l$=INKEY$: IF l$="0" TH
EN CLS : RANDOMIZE USR 50000: GO
 TO 2405
7098 BEEP .01,40
7100 LET z=H(j)/.95
7105 IF y=40 THEN LET z1=z: GO S
UB 7190
7110 IF y=20 THEN LET z1=1.1*z:
GO SUB 7190
7115 IF y=0 THEN LET z1=.6*z: GO
 SUB 7190
7120 NEXT y
```

```
7125 NEXT x
7130 RETURN
7135 REM
           - h 3d caroiaj -

7140 FOR x=0 TO 96 STEP 8
7145 PLOT cx+x,(cy+x/2)
7150 DRAW INK 7;-40,20
7155 NEXT x
7160 FOR y=0 TO 40 STEP 20
7165 PLOT cx-y,cy+y/2
7170 DRAW INK 7;96,48
7175 NEXT y
7180 RETURN
7185 REM
           - h 3d trasare -

7190 FOR k=0 TO z-1
7195 PLOT INK 7;cx+x-y,(cy+x/2)+
y/2+k
7200 DRAW INK 7;dx,dy
7205 NEXT k


7210 INVERSE 1: OVER 0
7215 PLOT INK 0;cx+x-y,(cy+x/2)+
y/2
7220 DRAW INK 7;dx,dy: DRAW INK
7;0,z: DRAW INK 7;-dx,-dy: DRAW
INK 7;0,1: DRAW INK 7;dx,dy: DRA
W INK 7;-dx,-dy: DRAW INK 7;0,-z
7225 DRAW INK 7;1,0: DRAW INK 7;
0,z1: DRAW INK 7;0,-z1
7230 INVERSE 0
7235 DRAW INK 7;1,0: DRAW INK 7;
0,z1+1: DRAW INK 7;-2,0: DRAW IN
K 7;0,-z1-1
7240 INVERSE 0
7245 RETURN
7500 REM
        --- List. auto ---

7501 PRINT AT 0,0;"   Auto   Opt
iuni   Hist      "
7502 PAUSE 0
7503 BEEP .07,35: PAUSE 10: BEEP
 .07,35
7505 PRINT OVER 1;AT 0,2;"██████
"
7510 PRINT PAPER 5;AT 1,3;"
      ";AT 2,3;"█B-benzina   ";A
T 3,3;"             ";AT 4,3;"█M-
motorina ";AT 5,3;"            "
;AT 6,3;"█P-petrol    ";AT 7,3;"
         "
7515 LET l$=INKEY$
7520 IF l$="b" THEN LET p$="  BE
NZINA ": LET sel=1: GO TO 7537
7525 IF l$="m" THEN LET p$=" MOT
ORINA ": LET sel=2: GO TO 7537
7530 IF l$="p" THEN LET p$="  PE
TROL  ": LET sel=3: GO TO 7537
7535 GO TO 7515
7537 LET ord=t(sel)
7540 BEEP .07,35: PAUSE 10: BEEP
 .07,35
7541 PRINT PAPER 2; INK 7;AT 4,9
;"                   ";AT 5,9;"
```

```
 Nr   Nr. auto   Cap  ";AT 6,9;"
"
7542 FOR i=1 TO ord
7543 IF i>=13 THEN GO TO 7550
7545 PRINT PAPER 2; INK 7;AT i+6
,9;" ";i;"  ";AT i+6,12;" ";g$(s
el,i);AT i+6,23;" ";v(sel,i);" "
;AT i+6,27;"t";" "
7546 IF v(sel,i)=9 THEN PRINT PA
PER 2;AT i+6,26;" "
7547 NEXT i
7550 PRINT PAPER 2;AT i+6,9;"____
_____________________"
7555 PRINT PAPER sel; BRIGHT 1;
INK 7;AT i+6,5;"              ";AT i
+7,5;p$;AT i+8,5;"              "
7560 BEEP .07,35: PAUSE 10: BEEP
 .07,35
7565 PAUSE 0: CLS : BEEP .02,40:
 PAUSE 1: BEEP .04,40
7570 RETURN
7575 STOP
7700 REM
       --- List. livrari ---
7701 PRINT AT 0,0;"  Auto   Opt
iuni    Hist         "
7702 PAUSE 0
7705 BEEP .07,35: PAUSE 10: BEEP
 .07,35
7715 PRINT OVER 1;AT 0,9;"
     "
7720 PRINT PAPER 5;AT 1,7;"
          ";AT 2,7;"  -optiune o
ra ";AT 3,7;"               ";AT
 4,7;"  -livr. azi     ";AT 5,7;"
                 ";AT 6,7;"  -livr.
 proc. ";AT 7,7;"
";AT 8,7;"  -livr lunare ";AT 9,
7;"                 ";AT 10,7;"  -
secventiere ";AT 11,7;"
     "
7725 LET l$=INKEY$
7730 IF l$="h" THEN GO TO 8100
7735 IF l$="a" THEN GO TO 7900
7740 IF l$="5" THEN GO TO 8700
7745 IF l$="l" THEN GO TO 8900
7747 IF l$="s" THEN GO TO 8800
7750 GO TO 7725
7755 RETURN
7900 REM
        --- Livrari azi ---
7905 PAUSE 0
7910 BEEP .07,35: PAUSE 10: BEEP
 .07,35
7915 PRINT PAPER 2; INK 7;AT 6,3
;"                ";AT 7,3;"  -benzi
na   ";AT 8,3;"               ";AT 9
,3;"  -motorina ";AT 10,3;"
       ";AT 11,3;"  -petrol    ";
AT 12,3;"               "
7920 PAUSE 0
7923 LET l$=INKEY$
7925 IF l$="b" THEN LET pr=3000:
 LET p$="   BENZINA   ": LET sel=
```

```
1: GO TO 7943
7930 IF l$="m" THEN LET pr=2000:
 LET p$="  MOTORINA  ": LET sel=
2: GO TO 7943
7935 IF l$="p" THEN LET pr=4000:
 LET p$="   PETROL   ": LET sel=
3: GO TO 7943
7940 GO TO 7923
7943 LET ord=t(sel)
7945 BEEP .07,35: PAUSE 10: BEEP
 .1,35
7950 PRINT PAPER 1; INK 6;AT 2,1
4;"            ";AT 3,14;"
            ";AT 4,14;"
"
7955 FOR i=1 TO ord
7956 IF i>=13 THEN GO TO 7975
7958 PRINT PAPER 1;AT 4+i,14;"
            "
7960 PRINT PAPER 1; INK 6;AT 4+i
,14;" ";i;AT 4+i,17;"";v(sel,i)
;AT 4+i,22;"tone "

7965 LET sum=sum+v(sel,i)
7970 NEXT i
7975 PRINT PAPER 1;AT 4+i,14;"
            "
7978 PRINT PAPER 1;AT 5+i,14;"
            "
7980 PRINT PAPER 1; INK 6;AT 5+i
,14;"Total ";sum;AT 5+i,24;" t."
7985 PRINT PAPER 1;AT 6+i,14;"
            "
7990 PRINT PAPER 1; INK 6;AT 6+i
,14;" Pret ";pr*sum/1000;"  "
7995 PRINT PAPER 1; INK 6;AT 7+i
,14;"     mii lei ";AT 8+i,14;"
            "
8000 PRINT PAPER 7; BRIGHT 1; IN
K 1;AT 5+2*sel,3;p$
8005 BEEP .1,26
8010 PAUSE 0: CLS : RETURN
8100 REM
     --- Optiune ora ---
8105 PAUSE 0: BEEP .07,35: PAUSE
 10: BEEP .07,35
8110 PRINT BRIGHT 1;AT 4,0;"
                 ";AT 5,0;" 0 or
a init. 7:00 AM ";AT 6,0;"
                 ";AT 7,0;"    aleg
eti ora ...  ";AT 8,0;"
            "
8115 LET l$=INKEY$
8120 IF l$="0" THEN GO TO 8160
8125 IF l$="x" THEN GO TO 8135
8130 GO TO 8115
8135 BEEP .07,35: PAUSE 10: BEEP
 .07,35: PRINT PAPER 6; BRIGHT 1
;AT 8,10;"            ";AT 9,
10;"  ora    min.  ";AT 10,10;"
            ";AT 11,10;"
            ";AT 12,10;"
            "; FLASH 1;AT 11,11;" "
8136 FOR s=1 TO 5: LET a=SQR s:
NEXT s
8137 LET l$=""
```

```
8138 LET k$=INKEY$
8139 IF k$="" THEN GO TO 8138
8140 FOR s=1 TO 3: LET a=SQR s:
NEXT s: LET l$=l$+k$
8141 PRINT FLASH 0; PAPER 6; BRI
GHT 1;AT 11,11;l$: BEEP .07,4
8142 IF LEN l$<2 THEN GO TO 8138
8143 LET o=VAL l$
8150 PRINT FLASH 1;AT 11,19;"▚"
8151 LET l$=""
8152 LET k$=INKEY$
8153 IF k$="" THEN GO TO 8152
8154 FOR s=1 TO 3: LET a=SQR s:
NEXT s: LET l$=l$+k$
8155 PRINT FLASH 0; PAPER 6; BRI
GHT 1;AT 11,19;l$: BEEP .05,8
8156 IF LEN l$<2 THEN GO TO 8152
8157 LET m=VAL l$
8158 LET pre=(o-7)*4
8159 FOR k=1 TO pre
8160 BEEP .1,6: GO SUB 8500
8162 NEXT k

8167 RETURN
8168 BEEP .07,35: PAUSE 10: BEEP
 .07,35: REM LET o=7: LET m=0: L
ET pre=1
8170 RETURN
8200 REM
  --- Umplerea golirea cond ---

8205 INVERSE 1
8210 PAUSE 80
8215 PLOT 94,w(n)+7: DRAW 9,0: D
RAW 3,-3: DRAW 2,0: DRAW -2,1: D
RAW 3,0:
8220 PLOT 104,w(n)+8: DRAW 3,2:
DRAW 0,-1: DRAW 2,0: DRAW 0,-3:
PLOT 105,w(n)+6: PLOT 105,w(n)+8
: PLOT 106,w(n)+7: DRAW 1,0
8225 BEEP .1,0
8230 PLOT 110,w(n)+6: DRAW 0,2:
DRAW 1,-1: DRAW 0,9
8235 BEEP .1,5
8240 DRAW 0,11: BEEP .1,10: DRAW
 0,12: BEEP .1,15: DRAW 1,1: DRA
W 0,1: DRAW 3,3: DRAW 1,0: DRAW
1,1: DRAW 12,0: BEEP .1,20: DRAW
 12,0 : BEEP .1,25: DRAW 12,0: B
EEP .1,30: DRAW 12,0: BEEP .1,35
: DRAW 9,0
8245 BEEP .1,40
8250 DRAW 1,-1: DRAW 1,0: DRAW 3
,-3: DRAW 0,-1: DRAW 1,-1: DRAW
0,-9
8255 BEEP .1,45
8260 PAUSE 80
8265 INVERSE 0
8270 RETURN
8300 REM
      --- Semnal acustic ---

8310 FOR w=0 TO 5
8320 BEEP .07,20+RND*15
8330 NEXT w
8340 RETURN
8400 REM
```

```
     --- Out cisterna ---
8405 FOR i=15 TO 1 STEP -1
8415 PRINT AT 7*n-3,31-i;" ";AT
7*n-2,31-i;" ";AT 7*n-1,31-i;" "
;AT 7*n,31-i;" "
8420 PRINT AT 7*n-3,32-i;a$(1 TO
 i);AT 7*n-2,32-i;e$(1 TO i);AT
7*n-1,32-i;f$(1 TO i);AT 7*n,32-
i;d$(1 TO i)
8423 BEEP .12,-30+(RND*3)
8425 NEXT i
8430 PRINT AT 7*n-3,31;" ";AT 7*
n-2,31;" ";AT 7*n-1,31;" ";AT 7*
n,31;" "
8432 RETURN
8435 REM
     --- In cisterna ---

8437 LET j$=b$(1)+g$(n,t(n))+b$(
12 TO )
8440 FOR i=1 TO 16
8445 BEEP .07,-10+(RND*6)


8450 PRINT AT 7*n-3,32-i;a$(1 TO
 i);AT 7*n-2,32-i;j$(1 TO i);AT
7*n-1,32-i;c$(1 TO i);AT 7*n,32-
i;d$(1 TO i)
8455 NEXT i
8460 RETURN
8500 REM
     --- RND Capacit cist ---
      --- RND Nr. auto ---
8503 LET n=1+INT (RND*3)
8517 LET t(n)=t(n)+1
8520 LET i=1+INT (RND*6)
8525 LET v(n,t(n))=c(i)
8530 LET aut=1+INT (RND*18)
8535 LET g$(n,t(n))=STR$ (INT (1
5+(RND*20)))+"-"+h$(aut)+"-"+STR
$ (INT (1100+(RND*9800)))
8537 LET r(n)=v(n,t(n))/16: LET
y(n)=r(n): LET i(n)=0
8540 RETURN
8600 REM
      --- Umplerea rez. ---

8610 PAUSE 10
8615 GO SUB 8300
8620 PRINT AT n*7-4,1; FLASH 1;"
▚"
8625 PRINT AT 0,0; FLASH 1;"▚";
FLASH
,w(n)+3
8635 FOR Y=0 TO 35
8704 BEEP .07,Y: PLOT 23,w(n)+3+
Y: DRAW OVER 1;70,0
8655 PAUSE 4
8724 NEXT Y
8665 PRINT AT n*7-4,1; FLASH 0;"
 "
8735 PRINT AT 0,0;"  Auto   Opt
iuni   Hist   "
8675 PAUSE 30
8744 RETURN
8700 REM
     --- Livrari procente ---
```

```
       --- Livrari prezente ---
8702 FOR k=1 TO 16
8704 PRINT PAPER 2;AT 2+k,8;"
                        "
8706 PRINT PAPER 2; INK 7;AT 4,1
2;"Benz.  Motor.  Petr.";AT 5,15
;"%";AT 5,21;"%";AT 5,27;"%"
8715 NEXT k
8717 FOR k=1 TO 11
8720 PRINT PAPER 2; INK 7;AT 6+k
,9;i$(k)
8725 FOR j=1 TO 3
8730 PRINT PAPER 2; INK 7;AT 6+k
,9+j*6;a(k,j);"  "
8735 NEXT j: NEXT k
8740 BEEP .04,40: PAUSE 1: BEEP
.06,30: PAUSE 0: RETURN
8800 REM
        --- Secventiere ---

8810 BEEP .02,40: PAUSE 0: BEEP

.05,40: PAUSE 1: BEEP .02,32
8816 PRINT PAPER 1; INK 7;AT 10,
10;"                 ";AT 11,1
0;"■■-bucla se inchide";AT 12,10
;"  secvential prin   ";AT 13,10;
"actiunea tastei ■■■■";AT 14,10;"
                   ";AT 15,10;"■
■-bucla se inchide";AT 16,10;"
         automat   ";AT 17,10;"
                   "
8820 LET l$=INKEY$
8825 IF l$="v" THEN LET secv=1:
RETURN
8830 IF l$="a" THEN LET secv=0:
RETURN
8835 GO TO 8820
8840 RETURN
8900 REM
       --- Livrari lunare ---

8902 FOR k=1 TO 16
8904 PRINT PAPER 2;AT 2+k,8;"
                        "
8906>PRINT PAPER 2; INK 7;AT 4,1
2;"Benz.  Motor.  Petr.";AT 5,1
4;"m i l .   l e i"
8915 NEXT k
8917 FOR k=1 TO 11
8920 PRINT PAPER 2; INK 7;AT 6+k
,9;i$(k)
8925 FOR j=1 TO 3
8930 PRINT PAPER 2; INK 7;AT 6+k
,9+j*6;(INT (a(k,j)*9.27*10))/10
;"  "
8935 NEXT j: NEXT k
8940 BEEP .04,40: PAUSE 1: BEEP
.06,30: PAUSE 0: RETURN
8945 RETURN
```



SINTEZA 21

Complemente informatice: • BASIC (Istoric, extensii şi particularizări) • Sisteme de operare (UNIX, MS-DOS, CP/M ş.a.) • Microprocesoare (8, 16, 32 biţi) • 19 limbaje de programare (cuvinte cheie Ada, Algol,.., Prolog,.., Smalltalk) • Produse program (dBASE II, III,... Visicalc) • Vocabular comparativ al limbajelor de programare de nivel înalt (19+18 variante BASIC) şi al produselor program generalizabile (7-dBASE II, III, III Plus), Visicalc, Supercalc, Multiplan, Lotus 1-2-3, Symphony, Framework

□ BASIC-ISTORIC, EXTENSII, PARTICULARIZĂRI

**Arborele genealogic al limbajului BASIC...
pe o perioadă de 20 de ani**

Limbajul BASIC (Beginner's All-Pur-pose Symbolic Instruction Code, sau în traducere liberă, limbaj simbolic universal pentru începători) a fost creat în anul 1964 la Dartmouth College, Hanover, New Hampshire, S.U.A., de către un colectiv condus de prof. John G. Kemeny şi prof. Thomas E. Kurtz în scopul iniţierii în informatică a utilizatorilor fără experienţă în programarea calculatoarelor. BASIC 1 (prima versiune) a fost implementat pe calculatorul GE 225 (General Electric 225). La scurt timp după apariţia sa, limbajul BASIC a cunoscut o evoluţie impresionantă fiind curtat de numeroşi constructori de calculatoare (C.I,I., I.B.M., C.D.C., H.B., H.P., Logabax, Xerox, Philips etc.). În ultimii zece ani, limbajul BASIC a fost introdus definitiv în toate sistemele time-sharing, în sistemele cu mini- şi microcalculatoare şi, în special, în calculatoarele personale.

Odată cu evoluţia BASIC-ului s-au constatat diferenţe majore între versiunile implementate pe diferite tipuri de calculatoare faţă de versiunea iniţială propusă de Kemeny şi Kurtz în anul 1964. Această tendinţă în evoluţia limbajului BASIC vine în contradicţie cu imperativele de standardizare cerute de piaţă, de masa tot mai largă de utilizatori direcţi ai tehnicii de calcul.

Activitatea de standardizare a limbajului a început prin anul 1974 dar, foarte curînd s-a constatat că acest standard (ANSI X 360–American National Standard for Minimal BASIC) defineşte, de fapt, un limbaj mult prea îngust. În anul 1984 ANSI a definitivat versiunea proiectului pentru un BASIC extins (ANSI X3J/84–26), care va fi aprobat pînă în anul 1986. Ultima versiune, deosebit de puternică, distanţează BASIC-ul de toate celelalte limbaje de programare şi-l face comparabil cu limbajul PASCAL.

Prezentăm în continuare arborele genealogic al BASIC-ului pe o perioadă de 20 de ani, în care se precizează atît versiunea cît şi tipul de calculator pe care aceasta a fost implementată.

1964

Dartmouth Basic 1
GE 225

1965

Dartmouth Basic 2
GE 265
Honeywell
Series 200 Basic

1966

Dartmouth Basic 3
GE Mark 1
Honeywell
GCOS11
Batch Basic
IBM CPS
Basic
University of California
Berkeley Basic

1967

Dartmouth Basic 4
GE 635
DEC Poly
Basic
NCR Century
Basic 1
Comshare Commander 1
Basic

1968

Dartmouth Basic 5
IBM Call 360 Basic
DEC TSS-8
Timesharing Basic
University of
Maryland 1100 Basic
Xerox Varian 620
Basic Basic

1969

HP 2000 Basic
Honeywell Series 400
Basic
NCR Century 100 Basic

1970

Dartmouth Basic 6
Honeywell Series 600
Basic
DEC PDP-10 Basic
Unicomp 16 Basic 18 Basic

1971

DEC 8K PDP-8 Basic
Sperry/Univac UBasic
CDC Basic 2,0
Kronos
Varian V-73 Basic

1972

Data General Basic
DEC PDP-11 RSX Basic
HP 9830 Basic
Microdata Basic
Wang 2200 Basic

1973

HP 3000 Basic
HP 9831 Basic
Comshare * Basic

1974

Honeywell Level 6 Basic

1975

IBM S110 Basic
Altair/Microsoft 4K Basic
Altair/Microsoft 8K Basic
CDC NOC Basic
Gordon Eubanks E Basic

1976

Microsoft Extended Basic
Texas Instruments Internal
Standards
DEC Basic Plus 2
HP 1000D Basic
Comshare Commander 2
Basic 1976

1977

Apple Applesoft
Basic
Commodore
Basic
Tandy TRS-80
Basic
Microsoft Disk
Basic
Texas Instruments
DX 990 Basic
HP 9845
Basic
Digital Research
C Basic

1978

Microsoft
R-MAX Basic
HP 300 Basic
Honeywell
CP-6 Basic
Digital Research
C Basic II

1979

Dartmouth Basic 7
Honeywell DBS 6
Basic
Tandy TRS-80
Level-III Basic
Microsoft M Basic
(BASIC-80)
Texas Instruments
99/4a Basic
DECVAX Basic

1980

HP 1000L Basic
HP 85 Basic
IBM System 38 Basic
IBM System 34 Basic
IBM System 23 Basic

1981

Microsoft MS Basic
Microsoft Advanced
Basic
HP Series 200 Basic
IBM RM Basic
IBM Displaywriter Basic
Digital Research
CB-80 Basic

1982

Dartmouth
Basic 8
MSX Basic
Microsoft
GW Basic 1.0
Microsoft
Xenix 68000 Basic
Microsoft
Xenix 8086 Basic
HP 520 Basic
Digital Research
CB-86 Basic

1983

Commodore
Simon's Basic
Tandy
Model 100 Basic
Microsoft
GW Basic 2.0
Digital Research
P Basic
IBM System 36
Basic
Digital Research
CB-68 Basic

1984

True Basic
Apple Mac Basic
Morgan Computing
Professional Basic
Microsoft Macintosh
Digital Research

Şi în ultimii 5 ani limbajul a evoluat, în mod deosebit, datorită dezvoltării de noi microprocesoare, de noi calculatoare personal-profesionale, de noi aplicaţii, cu cerinţe speciale de rapiditate a răspunsului, de noi domenii – ca al inteligenţei artificiale – etc.

Remarcăm versiuni ca TURBO BASIC sau QUICK BASIC [63, 64], foarte rapide compilatoare cu facilităţi de grafică, de sunet, gestiune a discurilor Whinchester etc., ca şi **BASICA 2.0, de pe IBM PC şi de pe Olivetti 24 [65].**

Extensii sau particularizări BASIC:
MBASIC 86, ZBASIC, CBASIC, S-BASIC, BASIC-APPLESOFT, BASIC-ATARI, BASIC-TI, BETA BASIC

MBASIC 86

BASIC Microsoft 86 este o versiune extinsă a lui BASIC Microsoft 80. MBASIC 80 a fost scris pentru procesoare de 8 biţi, în timp ce MBASIC 86 este o extindere a lui MBASIC 80 pentru procesoarele de 16 biţi. MBASIC 86 nu conţine instrucţiuni grafice în schimb ABASIC (scris pentru IBM PC) şi ZBASIC (scris pentru Zenith Z100) – două dialecte MBASIC, adaugă aceste instrucţiuni.

Cuvinte rezervate

ABS, AND, ASC, ATN, AUTO

CALL, CDBL, CHAIN, CHR$, CINT, CLOAD, CLOAD?, CLOAD *, CLOSE, CLS, COMMON, CONT, COS, CSAVE *, CSNG, CVD, CVI, CVS

DATA, DEF FN, DEFDBL, DEFINT, DEFSNG, DEFSTR, DEF USR, DELETE, DIM

EDIT, END, EOF, ERASE, ERL, ERR, ERROR, EXP

FIELD, FILES, FIX, FOR . . . NEXT, FRE

GET, GOSUB, GOTO

HEX

IF . . . THEN . . . ELSE, INKEY$, INP, INPUT, INPUT#, INSTR, INT

KILL

LEFT$, LEN, LET, LINE INPUT, LINE INPUT#, LIST, LLIST, LOAD, LOC, LOF, LOG, LPOS, LPRINT, LPRINT USING, LSET

MERGE, MID$, MKD$, MKI$, MKS$, MOD

NAME, NEW, NEXT, NULL

OCT$, ON ERROR GOTO, ON GOSUB, ON GOTO, OPEN, OPTION BASE, OR, OUT

PEEK, POKE, POS, PRINT, PRINT USING, PRINT# USING

RANDOMIZE, READ, REM, RENUM, RESET, RESTORE, RESUME, RESUME 0, RESUME NEXT, RIGHT$, RND, RSET, RUN

SAVE, SGN, SIN, SPACES$, SPC, SQR, STEP, STOP, STR$, STRING$, SWAP, SYSTEM

TAB, TAN, TROFF, TRON

USR

VAL, VARPTR, VARPTR#

WAIT, WHILE . . . WEND, WIDTH LPRINT, WRITE, WRITE#

Formatele instrucţiunilor cu exemple de utilizare sînt prezentate în [61].

ZBASIC

ZBASIC reprezintă o implementare a lui BASIC Microsoft pe calculatoarele personale ZENITH 100 și 150. Dialectul diferă de implementările obișnuite ale lui BASIC Microsoft pe calculatoarele personale, în special în ceea ce privește instrucțiunile grafice și de sunet care sînt specifice lui ZENITH 100 și 150.

Cuvinte rezervate

ABS, AND, ASC, ATN, AUTO

BEEP, BLOAD, BSAVE

CALL, CDBL, CHAIN, CHR$, CINT, CIRCLE, CLEAR, CLOSE, CLS, COLOR, COMMON, CONT, COS, CSNG, CSRLIN, CVD, CVI, CVS

DATA, DATE$, DEFDBL, DEF FN, DEFINT, DEFSEG, DEFSNG, DEFSTR, DEF USR, DELETE, DIM, DRAW

EDIT, END, EOF, ERASE, ERL, ERR, ERROR, EXP

FIELD, FILES, FIX, FOR ... NEXT, FRE, GET#, GET, GOSUB, GOTO

HEX

IF ... THEN ... ELSE, INKEY$, INP, INPUT, INPUT#, INPUT$, INSTR, INT

KEY, KEY OFF, KEY ON, KILL

LEFT$, LEN, LINE, LINE INPUT, LINE INPUT#, LIST, LLIST, LOAD, LOC, LOCATE, LOF, LOG, LPOS, LPRINT, LPRINT USING, LSET

MERGE, MID$, MKD$, MKI$, MKS$, MOD

NAME, NEXT, NEW, NULL

OCT$, ON ERROR GOTO, ON GOSUB, ON GOTO, OPEN, OPTION BASE, OR, OUT

PAINT, PEEK, POINT, POKE, POS, PRESET, PRINT, PRINT USING, PRINT# USING, PSET, PUT

RANDOMIZE, READ, REM, RENUM, RESET

Formatele instrucțiunilor, cu exemple de utilizare, sînt prezentate în [61].

CBASIC

CBASIC este un compilator BASIC (sub CP/M, CP/M concurent, CP/M 86) care și-a cîștigat o popularitate mai mare printre programatorii riguroși, deoarece este un limbaj structurat, mult mai apropiat în concepție de FORTRAN decît de BASIC.

Cuvinte rezervate

ABS, ASC, ATN

CALL, CHAIN, CHR$, CLOSE, COMMAND$, COMMON, CONCHAR%, CONSOLE, CONSTAT%, COS, CREATE

DATA, DEF, DELETE, DIM

END, EXP

FEND, FILE, FLOAT, FOR ... NEXT, FRE

GOSUB, GOTO

IF, IF . . END, INITIALIZE, INP, INPUT, INT, INT%

LEFT$, LEN, LET, LOG, LPRINTER

MATCH, MID$

ON, OPEN, OUT

PEEK, POKE, POS, PRINT, PRINT#, PRINT USING, PRINT USING#

RANDOMIZE, READ, READ#, REM, RENAME, RESTORE, RETURN, RIGHT$, RND

SADD, SAVEMEM, SGN, SIN, SIZE, SQR, STOP, STR$

TAB, TAN

UCASE$

VAL, VARPTR

WEND WHILE

Formatele instrucţiunilor cu exemple de utilizare sînt prezentate în [61].

S-BASIC

Ca şi CBASIC, S-BASIC este un compilator implementat însă numai pe calculatoarele personale Kaypro (pe 8 biţi) sub sistemul de operare CP/M. S-BASIC este un BASIC structurat şi, pentru cineva familiarizat cu BASIC-ul Microsoft obişnuit, această versiune hibrid a limbajului seamănă mai mult cu un COMAL sau un PASCAL decît cu un BASIC adevărat.

Spre deosebire de CBASIC, S-BASIC conţine proceduri de control pentru urmărirea execuţiei programului, ceea ce permite o depanare mult mai uşoară. S-BASIC, spre deosebire de "vărul" BASIC Microsoft, este un limbaj structurat în multe privinţe asemănător cu PL/I sau PASCAL.

S-BASIC nu solicită numere de linie pentru fiecare instrucţiune. La fel ca şi "stră-moşul" său, FORTRAN-ul, S-BASIC solicită numere de linie numai pentru acele linii care conţin instrucţiunile **GOTO** şi **GOSUB**. De notat că variabilele S-BASIC trebuie să fie declarate (cu instrucţiunea **VAR**), restricţie care face ca acest limbaj să semene cu PL/I.

Inalta structurare a lui S-BASIC oferă utilizatorilor un grad mare de flexibilitate în folosirea variabilelor şi a tipurilor de date.

Cuvinte rezervate

BASED, BEGIN

CASE, CHAR, COMMENT, COM

CONSOLE, CREATE

DO

ECHO, ELSE, EQV, EXECUTE

FFIX, FINT

IMP

LOCATE ... AT, LOCATION, LPRINTER

NUM$

PROCEDURE

REPEAT ... UNTIL

SIZE, STEP

TEXT

VAR

XOR

$CONSTANT, $INCLUDE, $LINES, $LIST, $LOADPT, $PAGE, $STACK, $TRACE

Formatele instrucţiunilor cu exemple de utilizare sînt prezentate în [61].

BASIC-APPLESOFT

BASIC-APPLESOFT este o implementare a lui BASIC Microsoft (conţine în plus instrucţiuni grafice pentru joasă şi înaltă rezoluţie) pe calculatoarele personale Apple II, II+, II e şi II c.

Cuvinte rezervate

ATN

CHR$, CLEAR, CONT, COS

DATA, DEF FN, DIM, DRAW

EXP

FLASH, FN, FRE(0)

GET, GOSUB, GOTO

HCOLOR, HGR, HGR2, HIMEM, HOME, HPLOT, HTAB

IF THEN, IF THEN GOTO, INPUT, INT, INVERSE

LEFT$, LOG, LOMEM

MID$

NEXT, NORMAL

ON GOSUB, ON GOTO, ONERR GOTO

POS(0)

READ, RECALL, RESTORE, RESUME, RIGHT$, ROT

SCALE, SHLOAD, SIN, SPEED, SQR, STOP, STORE, STR$

TAB(0), TAB, TAN

USR

VAL

WAIT

XDRAW

Formatele instrucţiunilor cu exemple de utilizare sînt prezentate în [61].

BASIC-ATARI

BASIC-ATARI este implementat pe calculatoarele personale Atari 400 şi 800.

Cuvinte rezervate

ADR

BYE

CLOG, CLR, COM

DEG, DRAWTO

ENTER

INT

LIST

LOCATE, LPRINT

NEXT

OPEN

PADDLE, POINT, POP, POSITION, PTRIG

RAD, REM

SAVE, SETCOLOR, STATUS, STICK, STRIG

TRAP

XIO 18, XIO 254, XIO 35, XIO 36, XIO

Formatele instrucţiunilor cu exemple de utilizare sînt prezentate în [61].

BASIC-TI (extins)

BASIC-TI şi BASIC-TI extins sînt implementate pe calculatoarele personale TI 99/4A.

Cuvinte rezervate

ACCEPT, AND

CALL, CALL CHARPAT, CALL CHARSET, CALL COINC, CALL COLOR, CALL DELSPRITE, CALL DISTANCE, CALL ERR, CALL FILES (2), CALL INIT, CALL LINK, CALL LOAD, CALL LOCATE, CALL MAGNIFY, CALL MOTION, CALL PATTERN, CALL PEEK, CALL POSITION, CALL SAY, CALL SPGET, CALL SPRITE, CALL VERSION

DISPLAY, DISPLAY USING

FOR ... NEXT

IF ... THEN

LET, LINPUT

MAX, MERGE, MIN

NEXT, NOT

OLD, ON BREAK NEXT, ON BREAK STOP, ON ERROR, ON ERROR STOP, ON GOSUB, ON GOTO, ON WARNING NEXT, ON WARNING PRINT, ON WARNING STOP, OPTION BASE ERROR, OR

PRINT#USING, PRINT USING

RETURN, RETURN NEXT, RETURN WITHOUT GOSUB, RPT, RUN

SAVE, SIZE, SUB, SUBEND, SUBEXIT

XOR

Formatele instrucţiunilor cu exemple de utilizare sînt prezentate în [61].

BETA BASIC

BETA BASIC este un BASIC-SPECTRUM însă mult mai puternic. BETA BASIC permite lucrul cu proceduri şi transmiterea unor tablouri ca parametri. BETA BASIC permite scrierea de proceduri care se autoapelează recursiv. Programarea structurată este foarte bine reprezentată prin instrucţiunile **DO, LOP, EXIT IF, WHILE** şi **UNTIL**. În instrucţiunea **IF ... THEN** se poate folosi şi **ELSE**. Cu **ON** se poate selecta un număr de linie sau o instrucţiune dintr-o listă ceea ce face posibilă o formă restrînsă a instrucţiunilor **CASE** sau **SELECT**. Rutina **PRINT** şi instrucţiunile grafice sînt mult mai complexe.

Cuvinte rezervate (BETA BASIC 3.0)

ALTER, AND, AUTO, ABS, ACS, ASN, AT, ATN, ATTR

BREAK, BIN$, BEEP, BORDER, BREAK, BRIGHT

CAT, CLEAR, CLOCK, CLS, COPY, CSIZE, CHAR$, COSE, CHR$, CIRCLE, CLOSE#, CODE, CONTINUE, COS

DEFAULT, DEF KEY, DEF PROC, DELETE, DO, DO WHILE, DO UNTIL, DPOKE, DRAW TO, DEC, DPEEK, DATA, DEFFN, DIM

EDIT, ELSE, END PROC, ERASE, EXIT IF, EOF, ENTER, EXP

FILL, FILLED, FLASH, FN, FOR

GET, GOSUB, GOTO

HEX$

IF ... THEN ... ELSE, INARRAY, INSTRING, ITEM, IN, INK, INKEY$, INPUT, INVERSE

JOIN TO

KEYIN, KEYWORDS

LENGTH, LET, LIST, LLIST, LIST DATA, LIST VAL, LIST VAL$, LIST DEF KEY, LIST FORMAT, LIST PROC, LIST REF, LOAD, LOCAL, LOOP, LEN, LINE, LLIST#, LN, LPRINT LPRINT#,

MEM, MEMORY, MOD, MERGE

NUMBER, NEXT, NOT, NEW

ON, ON ERROR, OVER, OR, OPEN#, OUT

PLOT, POKE, POP, PAPER, PAUSE, PEEK, PI, POINT,

PRINT, PRINT#

READ, READ LINE, REF, RENUM, ROLL, RNDM, REM, RESTORE, RETURN, RUN, RND, RANDOMIZE

SAVE, SAVE DATA, SCROLL, SORT, SPLIT, SCRN$, SHIFT$, SINE, STRING$, SCREEN$, SGN, SIN, SQR, STEP, STOP, STR$

TRACE, TIME$, TAB, TAN, THEN, TO

UNTIL, USING, USING$, USR

VERIFY, VERIFY DATA, VAL, VAL$

WHILE, WINDOW

XOS, XRG, YOS, YRG, XOR

☐ Sisteme de operare pe calculatoare medii-mari, pe mini-, micro-, pe calculatoare personal-profesionale și personale (UNIX, MS-DOS, CP/M, OS/2 ș.a.)

Evoluția sistemelor de operare

Odată cu dezvoltarea rapidă a microelectronicii, familiile de calculatoare și microprocesoare au cunoscut o largă diversificare iar sistemele de operare au înregistrat o dinamică fără precedent.

Atît sistemele de operare ROM (pentru calculatorul personal Commodore) cît și cele pe casetă magnetică (pentru Commodore, Radio Shack, Atari etc.) și pe disc magnetic (Disk Operating System) proiectate în vederea asigurării gestiunii optime a resurselor au venit în contradicție cu cerințele de standardizare solicitate de numărul tot mai mare de utilizatori ai tehnicii de calcul.

DOS-ul s-a impus ca o necesitate pentru majoritatea calculatoarelor personale. De notat că Apple a fost printre primele firme care au utilizat un sistem de operare DOS.

Heath/Zenith 89, Altos, North Star au folosit sistemul de operare CP/M de la Digital Research, unul dintre primele sisteme de operare DOS. În timp ce versiunile de început ale lui APPLE DOS erau strict specializate, sistemul de operare CP/M prezenta un înalt grad de portabilitate.

CP/M este încă considerat unul din standardele industriale, deși a început să fie înlocuit pe scară largă de către MS-DOS (sistem de operare creat pentru IBM PC), MS-DOS (compatibil) devenind un cuvînt de viitor pentru calculatoarele personale pe 16 biți.

CP/M și MS-DOS se comportă asemănător, utilizatorii lui CP/M găsind adesea că trecerea la MS-DOS nu este deloc șocantă. APPLE DOS (sistemul de operare cu disc APPLE) are caracteristici comune cu MS-DOS, ceea ce dezvăluie originile sale timpurii. De notat, însă, că sistemul de operare cu disc APPLE 3.3 (APPLE DOS 3.3) este înlocuit de un nou sistem de operare ProDOS, care a fost proiectat să funcționeze pe APPLE II e și II c.

Viitorul sistemelor de operare este legat însă de evoluția lui UNIX ce s-a impus în deceniul 8 ca principala soluție de standardizare a sistemelor de operare pentru toate tipurile de calculatoare (maxi, mini, micro, PC-uri). UNIX-ul cît și XENIX-ul sînt susținute de un număr considerabil de producători de software, dar și de entuziasmul fără precedent al utilizatorilor.

Un viitor cu totul deosebit se conturează pentru utilizatorii lui Apple's LISA și Macinstosh cît și pentru cei ai lui Xerox Star, al căror sistem de operare este bazat aproape în întregime pe imagine.

Deoarece sistemele hardware continuă să furnizeze memorii compactizate de dimensiuni din ce în ce mai mari, cu facilități de adresare mai complexe, proiectanții de sisteme de operare pot aloca mai mult spațiu pentru software-ul necesar "gospodăririi"

funcţiunilor direct sub controlul utilizatorului. Rezultatul va fi o îmbinare a software-ului de aplicaţii cu sistemele de operare coordonate şi integrate, similar cu programul TOPVIEW realizat de firma IBM.

UNIX

Sistemul de operare UNIX a fost dezvoltat la mijlocul anilor '70 de către Ken Thompson şi Dennis Ritchie de la Bell Laboratories, AT&T S.U.A. Sistemul a fost o sinteză între multe idei bune existente atunci şi cîteva idei noi. Versiunea iniţială a fost mai ales elegantă şi prezenta o unitate a proiectării rar întîlnită în sistemele de dimensiuni mai mari.

Trei sînt caracteristicile mai importante ale sistemului de operare UNIX: **simplitate în proiectare, apropierea de un instrument software şi o deosebire minimă între "sistem" şi "utilizatorul" programelor.**

Ca o completare la caracteristicile sale generale amintim: **capacitatea de multiprelucrare, capacitatea de "multiuser", portabilitatea, acceptarea aproape universală în educaţie etc.**

Puţin istoric. Sistemul UNIX a fost mai întîi utilizat la laboratoarele Bell (cu regim de uz intern), fiind apoi îmbunătăţit treptat. El a intrat în atenţia mai multor universităţi, fiind lansat fără sprijin şi fără nici o reţinere în scopuri educative. În anul 1975 se poate vorbi de prima versiune comercială V6 (la momentul respectiv nu exista nici o versiune UNIX) destinată procesorului DEC PDP-11. Această versiune a cunoscut o răspîndire rapidă în cîteva universităţi şi a început să fie utilizată în proiecte comerciale şi de conducere. AT&T a folosit, de asemenea, versiunea a şasea ca punct de plecare pentru alte versiuni derivate UNIX.

Versiunea 7 a lui UNIX a fost distribuită gratuit în universităţile din S.U.A. în perioada 1978–1979, fiind destinată calculatoarelor DEC PDP-11, DEC VAX 11/780. UNIX s-a răspîndit rapid în lumea universitară. Sistemul era elegant şi uşor de utilizat. Universităţile din California şi Berckley au realizat cele mai însemnate variante academice ale sistemului UNIX.

În paralel cu universităţile, AT&T a continuat propriile sale cercetări. Introducerea sistemului UNIX III în 1981–1982 pentru DEC VAX 11/780 (propus ca standard UNIX), răspîndirea sistemului UNIX (System III, V7) pe diferite tipuri de calculatoare şi/sau microprocesoare, în perioada 1982–1983, a constituit începutul unei perioade de vîrf pentru comercializarea activă a sistemului. Sistemul V a urmat sistemului UNIX III (sistemul IV nu a existat). Au fost rezolvate un număr mare de probleme ale sistemului şi a fost îmbunătăţită documentaţia. Au fost adăugate alte noi caracteristici, inclusiv capacitatea de comunicare între procese. Astăzi există o varietate de sisteme comerciale UNIX sub diverse denumiri comerciale.

Istoria numeroaselor versiuni UNIX a creat o industrie, care pune acum problema standardizării.

Comparaţii cu alte sisteme de operare. Sistemele de operare tradiţionale cu structură complexă sînt optimizate la un înalt nivel pentru structuri hardware particulare. Aceste sisteme tind să fie complexe şi greu de manevrat în comparaţie cu UNIX, fiind legate de configuraţia sistemelor de calcul respective.

În lumea microcalculatoarelor sistemul UNIX concurează cu alte sisteme cu firmă, cum sînt MS-DOS. În timp ce UNIX-ul dispune de un set de caracteristici mai avansate, MS-DOS-ul este pe undeva mai simplu în concepţie şi în cererea de resurse, ceea ce îl face mai accesibil pentru preţurile mai scăzute de pe piaţa de masă a microcalculatoarelor. MS-DOS-ul este portabil numai pentru anumite calculatoare, funcţie de arhitectura acestora. UNIX este portabil dincolo de arhitectură.

Sistemul UCSD P furnizează o altă comparaţie accesibilă. Sistemul P este portabil la un înalt nivel şi poate fi adaptat la o nouă configuraţie, cu eforturi mai mici decît UNIX. Pe de altă parte, sistemul P este orientat către echipamente cu preţ scăzut destinate unor utilizatori mai nepretenţioşi, în timp ce UNIX acoperă un domeniu mult mai mare, de la calculatoare personale pînă la maxicalculatoare.

Atît sistemul P cît şi MS-DOS-ul sînt optimizate pentru echipamente destinate unui singur utilizator, drept consecinţă, ele prezintă un grad mai înalt de sensibilitate, cu multe comenzi.

Sistemele UNIX solicită anumite capacităţi hardware (minimum 256 Ko memorie principială) restricţie ce împiedică folosirea lor şi pentru calculatoarele personale cele mai ieftine.

IBM PC-XT reprezintă un exemplu de calculator personal care poate utiliza relativ fără dificultăţi sistemul UNIX. Cele mai recomandabile sînt însă calculatoarele IBM PC-AT, sau cele pe 32 de biţi.

Concluzie. Caracteristica pieţei pentru UNIX se schimbă cu rapiditate. Sistemul trece de la utilizări specializate la utilizări generale, de la o raritate la un produs obişnuit, de la produs academic la produs comercial, de la programatori la utilizatori etc. AT&T şi numeroase alte firme au cooperat pentru a produce aşa-zisa "generic version" a lui UNIX pentru Intel, Motorola, National şi Zilog. UNIX a avut o puternică influenţă în proiectarea altor sisteme de operare. De exemplu, ultima versiune MS-DOS include un număr apreciabil de caracteristici derivate din UNIX.

Toate acestea demonstrează că succesul UNIX-ului se datorează atît condiţiilor economice favorabile, dar în special inteligenţei privind soluţiile tehnice adoptate.

U

U (marcă înregistrată de către ITCI România) este compatibil cu sistemul UNIX V7 — marcă înregistrată de AT&T, S.U.A. păstrînd caracteristicile de bază şi convenţiile specifice tuturor sistemelor de operare de tip UNIX. Scris în mare parte în limbajul C, sistemul U prezintă un grad înalt de portabilitate, oferind astfel un mediu integrat de utilizare a tehnicii de calcul.

Sistemul este operaţional din anul 1987 şi conţine instrumente pentru dezvoltarea de programe în limbajele C, FORTRAN 77, BASIC, PASCAL, PROLOG.

Sistemul conţine un nucleu [62] (partea rezidentă) şi programe utilitare accesibile prin interpretorul de comenzi (Shell). Spre utilizator sistemul prezintă următoarele nivele de interfeţe: un nivel exterior nucleului (oferit de utilitare), un nivel intermediar (oferit de biblioteca standard C) şi un nivel scăzut (oferit de directivele de sistem).

MIX

MIX este numele sistemului de operare elaborat în cadrul ICSIT-TCI pentru gama minicalculatoarelor româneşti din familiile INDEPENDENT şi CORAL.

Există mai multe versiuni ale acestui sistem, ultima fiind versiunea MIX / PLUS. MIX / PLUS respectă structura standard a sistemului de operare implementat pe minicalculatoarele compatibile cu sistemul de operare RSX-11M elaborat de firma DEC (Digital Echipament Corporation). Sub MIX pot fi utilizate compilatoarele : COBOL, FORTRAN 77, PASCAL, PROLOG. Tot sub MIX există programe de bază specializate în: gestiune colecţii de date organizate în fişiere (programul SIRIUS-100 compatibil cu programul DATATRIEVE), gestiunea bazelor de date (SGBD-urile ARGUS, LEDA, MIDAS, RECOL, ORACLE etc.). Sistemul de operare MIX pune la dispoziţia utilizatorilor următoarele programe utilitare: EDI, EDT, RUNOFF (editoarele de texte/documente); PIP, BRU, FLX, CNV, SRT etc., bibliotecarul pentru gestionarea bibliotecilor de programe. Sînt şi programe care oferă facilităţi pentru lucru cu formate de ecran (programul SIPET) pentru reprezentarea grafică a informaţiilor economice (programul MANGRAF) etc.

MS-DOS

Sistemul de operare MS-DOS a fost dezvoltat iniţial de Microsoft pentru IBM-PC bazat pe microprocesorul 8088. Cînd este implementat pe IBM, sistemul de operare se numeşte PC-DOS sau mai simplu DOS. Atunci cînd este implementat pe alte calculatoare personale compatibile el se numeşte MS-DOS. MS-DOS a fost lansat în versiunile 1.0 (pentru calculatorul iniţial), 2.0 pentru XT, 2.1 pentru PC jr, 3.0 pentru AT ş.a.m.d. De notat că fiecare modernizare a lui MS-DOS a adăugat extinderi şi comenzi noi cu menţinerea comenzilor variantei precedente.

DOS-PC

Sistemul de operare DOS-PC a fost implementat la noi în ţară pe calculatoarele de fabricaţie românească: M216, FELIX PC, JUNIOR PC şi DACICC-20 PC. DOS-PC este compatibil cu sistemele de operare MS-DOS, PC-DOS şi este utilizat pe calculatoarele de tip IBM PC (XT sau AT), IBM PS/2 sau altele compatibile cu acestea.

Versiuni DOS-PC. Structura şi conceptele de bază ale sistemului DOS-PC se regăsesc în versiunile V2.00 şi V2.10. Cea mai nouă versiune a sistemului DOS-PC este V3.30 şi este descrisă în [60].

Facilităţi introduse de versiunea V3.00. Comenzile noi introduse sînt: **ATTRIB, LABEL, SHARE, DEVICE, FCSS, LASTDRIVE.** S-au adus îmbunătăţiri comenzilor: **FORMAT, BACKUP, RESTORE, DISKCOMP** şi **DISKCOPY.**

Facilităţi introduse de versiunea V3.10 [60]. Comenzile noi introduse sînt: **JOIN, SUBST.** S-au adus îmbunătăţiri comenzilor **LABEL** şi **TREE.**

Facilităţi introduse de versiunea V3.30 [60]. Comenzile noi introduse sînt: **APPEND, FASTOPEN.** S-au adus îmbunătăţiri comenzilor: **ATTRIB, BACKUP, FDISK, RESTORE.**

Avantajele utilizării sistemului de operare DOS-PC sînt: mecanism evoluat de tratare şi detecţie a erorilor; interfaţă utilizator simplă, uşor accesibilă; modularitate; o structură de fişiere eficientă; contabilitatea datei şi a timpului; posibilitatea conectării unei game variate de echipamente periferice; diversitatea foarte mare a resurselor software (compilatoare – FORTRAN, FORTRAN 77, PASCAL, Turbo PASCAL, C, C++, Turbo C, COBOL, Turbo PROLOG, LISP 86, MODULA-2, ADA; interpretoare – BASIC; asambloare ASM, MASM; depanator – DEBUG; editor de legături – LINK; bibliotecar – LIB; editoare de text – WORDSTAR, EDIT, EDLIN; sisteme de gestiune a bazelor de date – dBASE II, dBASE III, dBASE III PLUS; programe de comunicaţii – KERMIT; utilitare – SPY, EXPLORER; generatoare de rapoarte, tabele – MULTIPLAN, LOTUS 1–2–3).

Lista comenzilor DOS-PC: APPEND, ASSIGN, ATTRIB, BACKUP, BREAK, CHDIR, CHKDSK, CLS, COMP, COPY, CTTY, DATE, DEL, DIR, DISKCOMP, DISKCOPY, ERASE, EXE2BIN, FASTOPEN, FIND, FORMAT, GRAFTABL, GRAPHICS, JOIN, LABEL, MKDIR, MODE, MORE, PATH, PRINT, PROMPT, RECOVER, RENAME (REN), REPLACE, RESTORE, RMDIR, SELECT, SET, SHARE, SORT, SUBST, SYS, TIME, TREE, TYPE, VER, VERIFY, VOL, XCOPY.

Formatele comenzilor cu exemple de utilizare sînt prezentate în [60].

Z-DOS

Sistemul de operare Z-DOS este o versiune a lui MS-DOS implementată pe calculatoarele ZENITH (Z-100).

Lista comenzilor Z-DOS: CHKDSK, CONFIGUR, COPY, CREF, DATE, DEBUG, DEL, DIR, DSKCOMP, DSKCOPY, EDLIN, EXE2BIN, FILCOM, FORMAT, LIB, LINK, MAKE, MAP, MASM, PAUSE, PRINT, RDCPM, REM, REN, SYS.

Formatele comenzilor sînt prezentate în [61].

COMMODORE-DOS

Sistemul de operare DOS pentru calculatorul Commodore este un sistem hibrid şi diferit de CP/M, MS-DOS sau APPLE DOS unde BASIC-ul este încărcat ca şi cum ar fi un fişier program. Deoarece pe calculatorul Commodore, BASIC-ul este rezident el nu mai trebuie să fie încărcat de pe o memorie externă. Prin urmare, multe din comenzile sistemelor de operare pentru Commodore sînt în realitate comenzi în limbajul BASIC.

Lista comenzilor COMMODORE-DOS pentru calculatoarele Commodore 64, VIC 20, PET: CLOSE (B), **CMD** (B), **COPY, GET#** (B), **INITIALIZE, INPUT#** (B), **LOAD** (B), **NEW, OPEN** (B), **PRINT#** (B), **RENAME, RUN, RUN** (n), **SAVE** (B), **SCRATCH, VALIDATE, VERIFY** (B).

Formatele generale ale comenzilor (comenzile accesibile prin BASIC au fost indicate prin B în interiorul parantezei) sînt prezentate în [61].

APPLESOFT DOS 3.3

Apple DOS, spre deosebire de CP/M este un sistem de operare orientat către utilizator. DOS-ul lui Apple a fost scris în special pentru a simplifica procesul complex de creare și de acces al fișierelor.

Deși DOS-ul Apple a fost acum înlocuit pe scară largă de Pro DOS (pe II C și II e) contribuția sa la dezvoltarea calculatoarelor personale nu poate fi dată niciodată uitării.

Lista comenzilor Applesoft DOS 3.3: APPEND, B, BLOAD, BRUN, BSAVE, C, CALL, CATALOG, CHAIN, CLOSE, DELETE, EXEC, FP, GET, HIMEM, IN, INIT, INPUT, LOAD, LOCK, MAXFILES, MON, MON C, MON I, MON O, NOMON, OPEN, POSITION, PR, RWTS, READ, RENAME, RUN, SAVE, TRACE, UNLOCK, VERIFY, WRITE.

Formatele generale ale comenzilor sînt prezentate în [61].

ProDOS

ProDOS (Professional Disk Operating System) este un sistem de operare dezvoltat pe calculatoarele Apple II e și II c, ProDOS seamănă cu MS-DOS 2,0 al lui Microsoft și versiunile mai tîrzii ale lui UNIX. ProDOS este unic ca sistem de operare fiind mai degrabă un meniu cu mai multe funcțiuni.

Lista comenzilor și formatele generale asociate sînt prezentate în [61].

TRS-DOS

TRS-DOS asemenea lui Apple DOS și CP/M a fost unul din primele sisteme de operare dezvoltate pentru calculatoarele personale. Deși TRS-DOS este sistemul de operare cu disc al lui Radio Shack, el cuprinde totuși multe din comenzile întîlnite la sistemele de operare Apple și CP/M.

Lista comenzilor TRS-DOS: AGAIN, ANALYZE, APPEND, ATTRIB, AUTO, BACKUP, BUILD, CLEAR, CLOCK, CLOCK OFF, CLS, COPY, CREATE, DATE, DEBUG, DIR, DO, DUAL, DUMP, ECHO, ERROR, FORMAT, FORMS, FREE, HELP, HOST, HOST OFF, I, KILL, LIB, LIST, LOAD, MOVE, PAUSE, PRINT, PROT, PURGE, RECEIVE, RENAME, RESET, SCREEN, SETCOM, SPOOL, STATUS, T, TIME, VERIFY, VERIFY OFF.

Formatele generale ale comenzilor sînt prezentate în [61].

TRS-DOS 6.0

Cunoscut fie ca L-DOS sau ca TRS-DOS 6.0, versiunea cea mai recentă a lui Tandy pentru sistemul de operare destinat lui TRS-80 oferă utilizatorului mult mai multe facilități privind crearea și exploatarea fișierelor.

Lista comenzilor TRS-DOS 6.0 operațional pe TRS-80 Model IV: APPEND, ATTRIB, AUTO, BACKUP, BOOT, BUILD, CLEAR, CLOCK, CLOCK OFF, CLS, COMM, CONV, CPY, CREATE, DATE, DEBUG, DEVICE, DIR, DO, DUMP, ECHO, ERROR, FILTER, FORMAT, FORMS, LOAD, MEMORY, PATCH, PRINT, PURGE, REMOVE, RENAME, RESET, ROUTE, RUN, SET, SETCOM, SETKI, SPOOL, SYSGEN, SYSTEM, TAPE, TIME, VERIFY, VERIFY OFF.

Formatele generale ale comenzilor sînt prezentate în [61].

Familia CP/M

CP/M a fost unul din primele sisteme de operare complete dezvoltate mai întîi pentru microprocesorul 8080 apoi pentru Z80.

În anul 1974 Garry Kildall, funcționar la firma INTEL, lucra pentru punerea la punct a unui program utilitar destinat supervizării funcționării dischetelor calculatoarelor per-

sonale. Pentru comercializarea produsului său, Kildall a fondat societatea Digital Research, care în anii 1981–1982 a cunoscut un mare succes.

CP/M este structurat în cinci zone distincte: **Zona sistemului (PAGE 0), zona TPA (Transian Area), zona CCP (Console Command Processor), zona BDOS (Basic DISK Operating System) și zona BIOS (Basic Input Output System).** Ultimele două zone (BDOS și BIOS) sînt numite FDOS (Funcțional DISK Operating System) sau altfel spus ansamblul de comenzi de gestiune al sistemului. Ultimele trei zone (CCP, BDOS, BIOS) formează CP/M-ul propriu-zis.

Versiuni CP/M

CP/M a cunoscut numeroase modificări care s-au concretizat prin următoarele numere de versiuni:

- **CP/M 80,** versiune DOS pentru 8 biți cu numerele de versiuni 1.3 – prima versiune comercializată în anul 1985 (practic, ea a dispărut); 2 și 2.2 (versiunea actuală pentru numeroase calculatoare personale pe 8 biți – ele sînt din ce în ce mai rare); versiunea 3.0, cunoscută și sub numele de CP/M Plus este ultima versiune CP/M pentru microprocesorul Z80 (versiunea echipează calculatoarele AMSTRAD, COMMODORE etc.).
- **MP/M,** versiune multiutilizator.
- **CP/M86,** versiune pentru 16 biți, cea mai puternică versiune a sistemului de operare CP/M. Ea a avut mai puțin succes ca CP/M 80 și este disponibilă ca opțiune pentru calculatoarele IBM-PC și cele compatibile cu acesta. Versiunea CP/M 86 a fost scrisă pentru microprocesoarele 8088 și 8086 care echipează marea majoritate a calculatoarelor personale de mică gestiune.
- **CCP/M** sau **CP/M concurent,** versiunea rețea pentru CP/M 86.

Tipuri de comenzi CP/M

Comenzile CP/M sînt de trei tipuri: comenzi rezidente, comenzi tranzitorii sistem, comenzi tranzitorii constructor.

Comenzile rezidente (R) sînt rezidente pe disc ca programe executabile (recunoscute de CCP). Comenzile CP/M rezidente sînt: **DIR, DIRSYS, ERASE, RENAME, TYPE, USER.**

Remarcă. Abrevierile uzuale ale comenzilor rezidente sînt următoarele: **DIRS (DYRSYS), ERA (ERASE), REN (RENAME), TYP (TYPE), USE (USER).**

Comenzile tranzitorii sistem (T) sau comenzile nerezidente – opera lui Digital Research – sînt comune tuturor calculatoarelor sub CP/M Plus; ele pot fi deci considerate ca făcînd parte din sistemul CP/M. Comenzile tranzitorii sînt: **DATE, DEVICE, DIR, DUMP, ED, ERASE, GBT, HELP, INITDIR, PATCH, PIP, RENAME, SAVE, SET, SHOW, STAT, SUBMIT, SID.**

Comenzile tranzitorii constructor (TC), foarte utile, dacă nu indispensabile, sînt specifice unui anumit tip de calculator. Ele sînt în general legate de aspectul hardware al calculatorului ca de exemplu: formatarea unei dischete sau copierea, pistă cu pistă a unei dischete.

Observație. Comenzile specifice calculatoarelor personale **AMSTRAD** sînt: **AMSDOS, DISKIT 3** (sub CP/M Plus), **DISKIT 2** (sub CP/M 2.2.), **LANGUAGE, FILECOPY** (sub CP/M 2), **PALETTE, SETKEYS, SETLST, SETSIO.**

Operațiile care se execută sub CP/M

Comenzile CP/M pot fi folosite pentru executarea mai multor operații după cum urmează.

OPERAȚII DE INIȚIALIZARE

- Obținerea datei și a orei
- Afișarea în permanență a datei și a orei
- Punerea la zi a datei și a orei
  - – direct
  - – prin dialog cu calculatorul
- Înlănțuirea automată a comenzilor
- Înlănțuirea comenzilor cu parametri

OPERAŢII ASUPRA PERIFERICELOR

- Lista unităţilor fizice ale sistemului
- Lista unităţilor asignate
  - cu CP/M2
- Lista completă a unităţilor
- Afişarea stării fizice/logice a ecranului/tastaturii
- Atribuirea unei unităţi logice la o unitate fizică
  - cu CP/M2
- Listarea caracteristicilor de afişaj ale ecranului
- Modificarea caracteristicilor de afişaj ale ecranului
- Schimbarea unei unităţi floppi disc

OPERAŢII ASUPRA DIRECTORY-ului

- Lista simplificată a fişierelor
  - de pe un alt disc
  - cu extensie particulară
  - cu un nume particular avînd primele caractere identice
- Lista atributelor unui fişier
  - cu CP/M2
- Pregătirea unui disc în vederea datării fişierelor
- Lista fişierelor cu data şi ora de creare
- Lista fişierelor cu atributul **DIR** sau **SYS**
- Lista fişierelor de pe toate cititoarele active sau de pe anumite cititoare
- Lista fişierelor cu toţi parametrii
- Lista fişierelor cu excluderea unora dintre ele
- Lista fişierelor în ordinea înscrierii lor în directory
- Lista fişierelor protejate sau neprotejate
- Lista fişierelor cu dimensiunile acestora
- Lista fişierelor
  - cu toate zonele utilizator
  - cu anumite zone utilizator
- Combinarea mai multor opţiuni

OPERAŢII DE BAZĂ CU FIŞIERE

- Listarea unui fişier text
- Listarea unui fişier în hexazecimal
- Ştergerea unui fişier
  - diferite fişiere
  - diferite fişiere cu confirmare
- Renumirea unui fişier
- Renumirea unui fişier (cu dialog)
- Renumirea globală a diferitelor fişiere

OPERAŢII DE TRANSFER ÎNTRE UNITĂŢILE PERIFERICE

- Utilizarea unui fişier pentru transmiterea de parametri unei comenzi
- Utilizarea unui fişier text pentru înlănţuirea comenzilor
- Copierea fişierelor de pe un disc pe altul
- Copierea unui fişier de pe un disc pe altul cu schimbarea numelui fişierului
- Copierea mai multor fişiere într-unul singur
- Crearea unui fişier text
- Copierea unui fişier disc pe o unitate fizică
- Dirijarea caracterelor emise de un periferic către un alt periferic
- Realizarea mai multor operaţii de transfer

OPŢIUNI

- Copierea unei liste de fişiere cu confirmare
- Copierea între două zone **USER**
- Copierea unei liste de fişiere fără afişarea numelui acestora
- Copierea unui fişier cu majuscule
- Copierea unui fişier cu numerotarea liniilor
- Copierea unei zone dintr-un fişier
- Copierea unui fişier cu verificarea copiei

INFORMAŢII OBŢINUTE DE PE DISC

- Spaţiul disponibil pe disc (cu CP/M2)
- Numele dischetei?
- Zonele **USER** folosite (cu CP/M2)
- Numărul de intrări directory disponibile
- Caracteristicile dischetei (cu CP/M2)
- Caracteristicile fizice ale unui fişier
- Caracteristicile fizice ale tuturor fişierelor (CP/M2)
- Facilităţile comenzii **STAT** (CP/M2)

UTILIZAREA ATRIBUTELOR FIŞIERELOR

- Numirea unui disc
- Fixarea unui cuvînt de trecere pentru fişiere
- Fixarea unui cuvînt de trecere pentru un disc
- Suprimarea unui cuvînt de trecere
- Activarea (dezactivarea) protecţiei prin cuvinte de trecere (cu CP/M2)
- Activarea (dezactivarea) unui disc numai pentru citire
- Modificarea atributelor directory ale unui fişier (cu CP/M2)
- Activarea (dezactivarea) funcţiei ARCHIVE
- Pregătirea discului pentru crearea fişierelor

FIXAREA DE VALORI PRIN LIPSĂ

- Unitatea de floppy disc
- Cuvîntul de trecere.

INIŢIALIZAREA SISTEMULUI

COPYSYS

COPYSYS (CP/M Plus; T)
A>**COPYSYS**
Funcţie: copiază sistemul de operare CP/M pe un disc. Comanda nu formatează discheta.

SUBMIT

FORMATUL 1
SUBMIT ⟨nume fişier⟩ (CP/M Plus, CP/M2; T)
A>**SUBMIT** DEBUT
Funcţie: lansează o secvenţă de comenzi.

FORMATUL 2
SUBMIT ⟨nume fişier⟩ ⟨parametru 1⟩ ⟨parametru 2⟩ ⟨parametru 9⟩ (CP/M plus, CP/M2, T)
A>**SUBMIT** LIST COM SUB
Funcţie: lansează o secvenţă de comenzi cu parametri (cel mult nouă).

DATE

FORMATUL 1
DATE (CP/M Plus; T)
A>**DATE**
Funcţie: obţine data şi ora.

FORMATUL 2
DATE (CP/M Plus; T)
A>**DATE C**
Funcţie: afişează data şi ora.

FORMATUL 3
DATE ZZ/LL/AA HH: MM: SS (CP/M Plus, CP/M2; T)
A>DATE 10/09/87 11 : 11 : 11
Strike Key to set time
Funcţie: afişează ora într-o singură operaţie.

FORMATUL 4
DATE SET (CP/M Plus; T)
A>**DATE SET**
Funcţie: iniţializează data şi ora prin dialog.

UTILIZAREA PERIFERICELOR

DEVICE

FORMATUL 1
DEVICE NAME (CP/M Plus; T)
A>**DEVICE NAME**
Funcţie: listează unităţile periferice ale sistemului de calcul.

FORMATUL 2
DEVICE VALUES (CP/M Plus; T)
A>**DEVICE VALUES**
Funcţie: listează unităţile periferice asignate.

FORMATUL 3
DEVICE (CP/M Plus; T)
A>**DEVICE**
Funcţie: generează lista completă a perifericelor.

FORMATUL 4
DEVICE CON (CP/M Plus; T)
DEVICE CRT (CP/M Plus; T)
A>**DEVICE CON**
A>**DEVICE CRT**
Funcţie: se obţine starea logică sau fizică a ecranului/claviaturii.

FORMATUL 5
DEVICE ⟨u.log : ⟩=⟨u.fiz⟩ (CP/M Plus; T)
A>**DEVICE CONOUT** : =LPT
Funcţie: reasignează o unitate logică la o unitate fizică.

FORMATUL 6
DEVICE CONSOLE [PAGE] (CP/M Plus; T)
A>**DEVICE CONSOLE** [PAGE]
Console Width set to 79 columns

Console page set to 24 lines
Funcţie: solicită caracteristicile de afişaj ale ecranului.

FORMATUL 7
DEVICE CONSOLE [**COLUMNS**]=⟨XX⟩
LINES=⟨YY⟩ (CP/M Plus; T)
A>**DEVICE CONSOLE** [**COLUMNS**=10
LINES=10]
Console Width set to 10 columns

LISTA FIŞIERELOR

DIR

FORMATUL 1
A>dir (CP/M Plus, CP/M2; R)
Funcţie: afişează lista simplificată a fişierelor de pe o dischetă.

FORMATUL 2
DIR ⟨C :⟩ (CP/M Plus, CP/M2; R)
unde ⟨C⟩ este numele cititorului de disc ce urmează a fi explorat.
A>**DIR** B :
Funcţie: afişează directory de pe un alt disc.

FORMATUL 3
DIR*.⟨extensie⟩ (CP/M Plus, CP/M2; R)
A>**dir***.com
Funcţie: afişează lista fişierelor cu o anumită extensie.

FORMATUL 4
DIR ⟨nume fişier*.*⟩
A>**DIR** D*.*
A : DISCKIT 3 COM : DATE COM:
DEVICE COM: DIR COM
A>
Funcţie: afişează lista fişierelor avînd aceleaşi caractere de început ale numelui fişierului ⟨nume fişier⟩.

FORMATUL 5
DIR [**ATT**] (CP/M Plus; T)
A>**DIR** [**ATT**]
Scanning Directory . . .
Sorting Directory . . .
Directory For Drive A : User 0

Name	**Bytes**	**Recs**	**Attributes**
AMSDOS COM	1 K	8	Dir RW
DATE COM	3 K	23	Dir RW

Funcţie: Afişează atributele fişierelor.

FORMATUL 6
DIR [**FULL**] (CP/M Plus; T)
A>**DIR** [**FULL**]
Funcţie: listează fişierele cu data la care au fost create.

FORMATUL 7
DIR (CP/M Plus, CP/M2; R)
A>**DIR**
Console page set to 10 lines
A>**DEVICE**
CONSOLE [C
COLUMNS=79
LINES=24
Console width set to 79 columns
Console page set to 24 lignes
Funcţie: modifică caracteristicile de afişaj ale ecranului.

Funcţie: listează fişierele cu atributul **DIR.**

FORMATUL 8
DIRS (CP/M Plus, CP/M2; R)
A>**DIRS**
A : SET COM: AMSDOS COM
NON – SYSTEM FILE (S) EXIST
A>
Funcţie: listează fişierele cu atributul **SYS.**

FORMATUL 9
DIR [DRIVE=ALL] (CP/M Plus; T)
A>**DIR** [drive=all]
Funcţie: listează fişierele de pe toate cititoarele active.

FORMATUL 10
DIR [ALL] (CP/M Plus, CP/M2; T)
Funcţie: listează fişierele cu toţi parametrii. Comanda este identică cu **DIR** [**FULL**].

FORMATUL 11
DIR ⟨nume fişier?*⟩
[**EXCLUDE**] (CP/M Plus; T)
A>**DIR*.COM** [**EXCLUDE**]
Scanning Directory . . .
Sorting Directory . . .
Directory For Drive A: User 0

Name	**Bytes**	**Recs**	**Attributes**
C10CPM3 EMS	25K	200	Dir RW

Funcţie: listează fişierele cu excluderea unora dintre ele.

FORMATUL 12
DIR [**NOSORT**] (CP/M Plus; T)
A>**DIR** [**NOSORT**]
Funcţie: listează fişierele în ordinea înscrierii lor în directory.

FORMATUL 13
DIR [**RW**] (CP/M Plus; T)
A>**dir** [rw]
Funcţie: listează fişierele protejate.

FORMATUL 14
DIR [**RO**] (CP/M Plus; T)
A>**dir** ⟨ro⟩
Funcţie: listează fişierele neprotejate.

FORMATUL 15
DIR [**SIZE**] (CP/M Plus; T)
A>**dir** [size]
Funcţie: listează fişierele cu capacitatea lor.

FORMATUL 16
DIR [**USER=ALL**] (CP/M Plus; T)
A>**DIR** [**USER=ALL**]
Funcţie: listează fişierele tuturor zonelor utilizator.

FORMATUL 17
DIR [⟨opt 1⟩ ⟨opt n⟩ . . .] (CP/M Plus; T)
A>**dir** [nosort ro]
Funcţie: combină mai multe opţiuni în cadrul comenzii **DIR.**

INITDIR

INITDIR ⟨n : ⟩ (CP/M Plus; T)
unde, ⟨n⟩ este numele cititorului pe care se găseşte discheta de iniţializat.
Funcţie: iniţializează un disc pentru datarea fişierelor.

OPERAŢII ELEMENTARE CU FIŞIERE

TYPE

TYPE ⟨nume fişier⟩ (CP/M Plus, CP/M2; R)
A>**TYPE** CPM.COP
Funcţie: listează conţinutul unui fişier text.

DUMP

DUMP ⟨nume fişier⟩ (CP/M Plus, CP/M2; T)
A>**DUMP** CPM .COP
Funcţie: listează conţinutul unui fişier în hexazecimal.

ERASE

FORMATUL 1
ERASE ⟨nume fişier⟩ (CP/M Plus, CP/M2; R)

FORMATUL 2
ERASE ⟨nume fişier⟩ (CP/M Plus, CP/M2; R)
A>**ERASE** RENAME.COM
Funcţie: şterge un fişier.

FORMATUL 3
ERASE ⟨nume fişier?✱⟩ (CP/M Plus, CP/M2; R)
A>**era**✱.TXT
Funcţie: şterge numai anumite fişiere.

FORMATUL 4
ERASE ⟨nume fişier?✱⟩ [C] (CP/M Plus; T)
A>**era** ✱.✱ [c]
A : LIST1 . (Y/N)? Y
A : LIST2 . (Y/N)? Y
A : PIP COM (Y/N)? n
A : ERASE . COM (Y/N)? n
A>dir
A : PIP COM : ERASE COM
A>
Funcţie: şterge fişierele cu confirmare pentru fiecare dintre ele.

REN

REN ⟨fişier 2⟩=⟨fişier 1⟩ (CP/M Plus, CP/M2; R)
A>**ren** copy.com=pip.com
Funcţie: renumeşte un fişier.

RENAME

FORMATUL 1
RENAME (CP/M Plus; T)
A>**RENAME**
Enter New Name : TEXTE1.TXT
Enter Old Name: TXT1.TXT
Funcţie: renumeşte un fişier pe bază de dialog (conversaţie).

FORMATUL 2
RENAME ⟨nume fişier?✱⟩=⟨nume fişier?✱⟩ (CP/M Plus, CP/M2; T)
A>**RENAME** TXT✱.TXT=FIC✱.TXT
Funcţie: renumeşte global diferite fişiere.

OPERAŢII DE TRANSFER ÎNTRE UNITĂŢILE PERIFERICE

GET

FORMATUL 1
GET FILE ⟨nume fişier⟩ (CP/M Plus; T)
A>**GET FILE** IN
Getting console input from file : IN
Funcţie: transmite automat argumentele unei comenzi.

FORMATUL 2
GET FILE ⟨nume fişier⟩ [**SYSTEM**] (CP/M Plus; T)
B>**GET FILE** LISTE [**SYSTEM**]
Getting console input from file : LISTE.
Funcţie: lansează o secvenţă de comenzi CP/M.

PUT

PUT CONSOLE FILE ⟨nume fişier⟩ (CP/M Plus; T)
A>**PUT CONSOLE FILE** LISTA
Putting console output to file : LISTA
Funcţie: stochează pe disc o ieşire ecran.

PIP

FORMATUL 1
PIP ⟨fişier 1⟩=⟨fişier 2⟩ (CP/M Plus; T)
A>**PIP** TEXTE2.TXT=TEXTE1.TXT
Funcţie: duplică un fişier pe acelaşi disc.

FORMATUL 2
PIP B : ⟨fişier⟩=A : ⟨fişier⟩
(CP/M Plus, CP/M2; T)
A>**PIP** B : TEXTE1.TXT=A : TEXTE1.TXT
Funcţie: copiază un fişier de pe un disc pe un altul cu un singur cititor. A şi B sînt numele celor două unităţi de discuri.

FORMATUL 3
PIP B : ⟨fişier 1⟩=A : ⟨fişier 2⟩
(CP/M Plus, CP/M2; T)
A>**PIP** B : TEXTE.BIS=A : TEXTE1.TXT
Funcţie: copiază un fişier de pe un disc pe altul cu schimbarea numelui.

FORMATUL 4
PIP ⟨nume fişier⟩=⟨nume fişier 1⟩, ⟨nume fişier 2⟩, ⟨nume fişier 3 . . .⟩
(CP/M Plus, CP/M2; T)
A>**pip** aga=ag1, ag2, ag3
Funcţie: regrupează mai multe fişiere într-unul singur.

FORMATUL 5
PIP ⟨fişier⟩=**CON** : (CP/M Plus, CP/M2; T)
A>**PIP** TEXT=**CON** :
Funcţie: crează un fişier text.

FORMATUL 6
PIP ⟨unitate fizică :⟩=⟨nume fişier⟩
(CP/M Plus, CP/M2; T)
A>**PIP** LST: = TEXT
Funcţie: copiază un fişier disc pe o unitate fizică.

FORMATUL 7
PIP ⟨unitate fizică 1 :⟩= ⟨unitate fizică 2 : ⟩ (CP/M Plus, CP/M2; T)
A>**PIP** LST: CON :
LLiiggnnee 11
LLiiggnnee 22
LLiiggnnee 33
A>
Ligne 1
Ligne 2
Ligne 3
Funcţie: transmite un caracter emis de un periferic către un altul.

FORMATUL 8
PIP (CP/M Plus, CP/M2; T)
A>**PIP**
CP/M3 PIP VERSION 3.0
*TEXTE1=texte
*TEXTE2=texte
Funcţie: lansează o secvenţă de transferuri succesive.

FORMATUL 9
PIP ⟨fişier 1⟩+⟨fişier 2⟩ [**C**] (CP/M Plus; T)
A>**PIP** B : = A : *.TXT [**C**]
Funcţie: copiază mai multe fişiere cu confirmare pentru fiecare.

FORMATUL 10
PIP ⟨fişier 1⟩[Ga]=⟨fişier 2⟩[Gb]
(CP/M Plus, CP/M2; T)
A>**PIP** TEXTE.TXT [G2]=TEXTE.TXT
Funcţie: copiază fişierele dintre două zone **USER.**

FORMATUL 11
PIP ⟨fişier?*1⟩=⟨fişier?*2⟩[**K**]
(CP/M Plus; T)
A>**pip** b:=*.TXT
A>**pip** b:=a:*.TXT [**K**]
Funcţie: copiază fără afişaj pe ecran.

FORMATUL 12
PIP ⟨fişier 1⟩=⟨fişier 2⟩[**U**]
(CP/M Plus, CP/M2; T)
A>**type** texte.txt
Ionel paşte vaca
A>**PIP** TEXTE.BIS=TEXTE.TXT [**U**]
A>
A>**TYPE** TEXTE.BIS
IONEL PAŞTE VACA
A>
Funcţie: copiază un text transformat în majuscule.

FORMATUL 13
PIP ⟨fişier 1⟩=⟨fişier 2⟩[**N**]
(CP/M Plus, CP/M2; T)
A>**PIP** TEXTE1.BIS=TEXTE1.TXT [**N**]
Funcţie: copiază un fişier numerotind liniile.

FORMATUL 14
PIP ⟨fişier 1⟩=⟨fişier 2⟩[**S**⟨şir caractere 1⟩ ↑ Z Q ⟨şir caractere 2⟩ ↑ Z]
(CP/M Plus, CP/M2; T)
A>**TYPE** COPYR
CP/M Version 3. 0 COPYRIGHT 1982, DIGITAL RESEARCH
A>**PIP** COPYR.BIS=COPYR [SCOPY ↑ ZQ 82 ↑ Z]
A>**TYPE** COPYR.BIS
COPYRIGHT 1982
A>
Funcţie: copiază numai o zonă dintr-un fişier.

FORMATUL 15
PIP ⟨nume fişier?*1⟩=⟨nume fişier?*2⟩ [**V**] (CP/M Plus, CP/M2; T)
A>**PIP** b:=texte.txt [**V**]
Funcţie: copiază fişierele, cu verificare.

OBŢINEREA INFORMAŢIILOR DE PE DISC

SHOW

FORMATUL 1
SHOW (CP/M Plus; T)
A>**SHOW**
A : RW, Space : 57 K
Funcţie: afişează spaţiul disponibil de pe dischete (RW pentru disc liber, RO dacă discul este folosit numai pentru citire).

FORMATUL 2
SHOW ⟨n:⟩ (CP/M Plus; T)
A>**show** b:
B : RW, Space : 120 K
Funcţie: afişează spaţiul disponibil de pe discul aflat pe cititorul ⟨n⟩.

FORMATUL 3
SHOW [**LABEL**] (CP/M Plus; T)
A>**SHOW LABEL**
Funcţie: permite vizualizarea numelui dat unei dischete prin comanda **SET**.

FORMATUL 4
SHOW [**USER**] (CP/M Plus; T)
A>**SHOW** [**USER**]
A : Active User : 0
Funcţie: afişează zonele **USER** folosite.

FORMATUL 5
SHOW [**DIR**] (CP/M Plus; T)
A>**SHOW** [**DIR**]
A : Number of free directory entries : 52
A>
Funcţie: afişează numărul disponibil de directory (fişiere).

FORMATUL 6
SHOW [**DRIVE**] (CP/M Plus; T)
A>**SHOW** [**DRIVE**]
Funcţie: afişează caracteristicile unei dischete.

FORMATUL 7
SHOW ⟨n:⟩[**DRIVE**] (CP/M Plus; T)
A>**SHOW** b: [**DRIVE**]
Funcţie: precizează caracteristicile unei dischete de pe unitatea ⟨n⟩.

STAT

FORMATUL 1
STAT (CP/M2)
A>**STAT**
A : RW, Free Space : 164 K
Funcţie: afişează spaţiul disponibil de pe o dischetă (R/W – pentru scriere/citire; R/O – pentru citire).

FORMATUL 2
STAT ⟨n:⟩ (CP/M2)
A>**STAT** A :
A : RW, Free Space : 164 K
A>
Funcţie: afişează spaţiul disponibil de pe o dischetă aflată pe unitatea ⟨n⟩.

FORMATUL 3
STAT ⟨n:⟩ **DSK**: (CP/M2)
A>**STAT** A:**DSK**:
Funcţie: obţine caracteristicile fizice ale unui disc.

FORMATUL 4
STAT ⟨nume fişier.extensie⟩ (CP/M2)
A>**STAT** ED.CMD
Funcţie: obţine caracteristicile fizice ale unui fişier.

FORMATUL 5
STAT VAL: (CP/M2)
A>**STAT VAL:**
STAT 2.2
Funcţie: obţine posibilităţile comenzii.

FORMATUL 6
STAT 488USR: (CP/M2)
A>**STAT USR:**
A : Active User : 0
A : Active Files : 0 5 15
A>
Funcţie: afişează zonele utilizator active.

FORMATUL 7
STAT 488DEV: (CP/M2)
A>**STAT DEV:**
CON : is TTY :
AXI : is TTY :
AXO : is TTY :
LST : is TTY :
Funcţie: cunoaşterea caracteristicilor fizice ale perifericelor.

FORMATUL 8
STAT ⟨n:⟩=R/O (CP/M2)
A>**STAT** A:=R/O
Drive A : set to Read Only (RO)
Funcţie: protejează un disc împotriva scrierii sau ştergerii.

FORMATUL 9
STAT ⟨nume fişier extensie⟩ **$SYS** (CP/M2)
A>**STAT** ED.CMD **$SYS**
A : ED .CMD set to System (Sys)
Funcţie: face ca un fişier să devină . . . opac în directory.

FORMATUL 10
STAT ⟨nume fişier . extensie⟩ **$DIR** (CP/M2)
A>**STAT** ED.CMD **$DIR**
Funcţie: face ca un fişier să devină accesibil în directory.

FORMATUL 11
STAT ⟨nume fişier. extensie⟩ **$R/O** (CP/M2)
A>
STAT ED.CMD **$R/O**
A : ED.CMD set to Read Only (RO)

Funcţie: protejează un fişier împotriva distrugerii.

FORMATUL 12
STAT ⟨nume fişier.extensie⟩ **$R/W** (CP/M2)
A>ERA ED.CMD
Bdos Err On A : File R/O
A>**STAT** ED.CMD **$R/W**
A : ED.CMD set to Read Write (RW)

MANIPULAREA ATRIBUTELOR FIŞIERELOR

SET

FORMATUL 1
SET [**NAME**=⟨nume⟩] (CP/M Plus; T)
A>**SET** [**NAME**=CP/M]
ERROR : Unrecognized option
Option : M
Label for drive A:

Directory Label	Passwds Read	Stamp create	Stamp Access	Stamp Update
A : CP	off	off	off	off

A>**SET** [**NAME**=CP-M]
Funcţie: numeşte un disc.

FORMATUL 2
SET ⟨nume fişier?*⟩ [**PASSWORD**=⟨nume trecere⟩] (CP/M Plus; T)
A>**SET** TEXTE.TXT [**PASSWORD**=ABC PROTECT=READ]
A : TEXTE.TXT Protection=READ, Password=ABC
A>
Funcţie: dă un cuvînt de trecere pentru un fişier.

FORMATUL 3
SET [**PROTECT**=**ON**] (CP/M Plus; T)
A>**SET** [**PROTECT**=**ON**]
Funcţie: activează protecţia unui disc.

FORMATUL 4
SET [**PROTECT**=**OFF**] (CP/M Plus; T)
A>**SET** [**PROTECT**=**OFF**]
Funcţie: dezactivează protecţia unui disc.

FORMATUL 5
SET ⟨nume fişier⟩ [**PASSWORD**=] (CP/M Plus, CP/M2; T):
A>**SET** TEXTE.TXT[PA=
A: TEXTE .TXT
Password?
A : TEXTE .TXT Protection=NONE,
Password=
Funcţie: anulează cuvîntul de trecere al unui fişier.

Funcţie: "sparge" un fişier protejat la scriere.

FORMATUL 13
STAT ⟨periferic I/O:⟩=**CRT**: (CP/M2)
A>**STAT** CON : = **CRT** :
Funcţie: modifică caracteristicile fizice ale unui periferic.

FORMATUL 6
SET ⟨nume fişier⟩ [**PROTECTION**=⟨nn⟩] (CP/M Plus, CP/M2; T)
unde, nn : =**READ**|**WRITE**|**DELETE**
A>**SET** TEXTE.TXT **Protection**=**READ, Password**=CBA
A>**SET** TEXTE.TXT [**PR**=**WRITE**]
A : TEXTE.TXT
Password?
A : TEXTE.TXT Protection=**WRITE**
Funcţie: modifică nivelul de protecţie al fişierelor.

FORMATUL 7
SET [**RO**] (CP/M Plus; T)
A>**SET** [**RO**]
Drive A : set to Read Only [**RO**]
Funcţie: plasează un disc numai pentru citire.

FORMATUL 8
SET [**RW**] (CP/M Plus; T)
A>**SET** [**RW**]
Drive A : set to Read Write (RW)
Funcţie: eliberează un disc numai pentru citire.

FORMATUL 9
SET ⟨nume fişier?*⟩ [**SYS**] (CP/M Plus, CP/M2; T)
A>**SET***.COM [**SYS**]
Funcţie: modifică atributele directory ale unui fişier.

FORMATUL 10
SET ⟨nume fişier?*⟩ [**DIR**] (CP/M Plus, CP/M2; T)
A>**SET***.COM [**DIR**]
Funcţie: modifică atributele directory ale unui fişier.

FORMATUL 11
SET ⟨nume fişier?*⟩ [**ARCHIVE**=**ON**] (CP/M Plus; T)
A>
SET*.TXT [**ARCHIVE**=**ON**]
Funcţie: activează funcţia **ARCHIVE** (facilitate CP/M ce permite copierea numai a fişierelor care au fost modificate).

FORMATUL 12
SET ⟨nume fişier?*⟩

[ARCHIVE=OFF] (CP/M Plus; T)
Funcție: dezactivează funcția **ARCHIVE**.

FORMATUL 13
SET [⟨opțiune⟩=**ON**] (CP/M Plus, CP/M2; T)
A>**SET** [CREATE=ON]

Funcție: dă un nivel de datare a fișierelor. Opțiunile pot fi: **CREATE, UPDATE, ACCESS**; argumentul poate fi **ON** sau **OFF**.

FIXAREA VALORILOR PRIN LIPSĂ

SET

SET [**DEFAULT**=
⟨cuvînt de trecere⟩] (CP/M Plus; T)
A>**SET** TEXTE2.TXT [**PASSWORD**=TXT2 **PROTECT**=**READ**]
A : TEXTE2.TXT Protection=READ, Password=TXT2
Funcție: fixează un cuvînt de trecere prin lipsă.

SETDEF

SETDEF ⟨n:⟩ (CP/M Plus, CP/M2; T)
A>**SETDEF** B:
Drive Search Path:
1st Drive —B:
Funcție: numește un cititor de dischetă prin lipsă.

Cuvinte cheie CP/M

ACCESS (opțiune), **ALL** (opțiune), **ARCHIVE** (opțiune), **ATT** (opțiune), **AUXIN**: (opțiune), **AUXOUT**: (opțiune)

C (opțiune), **CON** :(opțiune), **CONIN** :(opțiune), **CONOUT** :(opțiune), **CONSOLE** (opțiune)

DATE (comandă), **DEFAULT** (opțiune), **DEVICE** (opțiune), **DIR** (comandă), **DIRS** (comandă), **DISPLAY** (opțiune), **DRIVE** (opțiune), **DUMP** (comandă)

ED (comandă), **ERASE** (comandă), **EXCLUDE** (opțiune)

FILE (opțiune), **FULL** (opțiune)

GET (comandă)

HELP (comandă)

INITDIR (comandă)

LABEL (opțiune), **LINK** (comandă), **LST** (opțiune)

MAC (opțiune)

NAME (opțiune), **NOSORT** (opțiune)

PAGE (opțiune), **PASSWORD** (opțiune), **PIP** (comandă), **PROTECT** (opțiune), **PUT** (comandă)

RENAME (comandă), **RMAC** (comandă), **RO** (opțiune), **RW** (opțiune)

SET (comandă), **SHOW** (comandă), **SID** (comandă), **SIZE** (opțiune), **STAT** (comandă), **SUBMIT** (comandă), **SYS** (opțiune), **SYSTEM** (opțiune)

TYPE (comandă)

USER (comandă), **USER** (opțiune)

VALUES (opțiune)

OS/2

OS/2 (Operating System/2) este noul sistem de operare destinat familiei de calculatoare personale IBM, PS/2 (Personal System/2). Pentru Model 30, Model 50, Model 60, Model 80 firma IBM, împreună cu Microsoft a introdus PC-DOS 3.3 un sistem de operare multitasking pentru calculatoare cu o memorie internă de 640 KO, funcționînd cu microprocesoarele 80 286 și 80 386. Noul sistem de operare OS/2 este tot un sistem multitasking, flexibil, puternic, cu multiple facilități ce poate lucra atît pe IBM PC AT, XT 286, cît și pe modelele 50, 60 și 80 ale familiei PS/2. Versiunea extinsă a sistemului de operare OS/2 – AIX (Advanced Interactive Executive Operating System) urmează a fi introdusă de firma IBM pentru limbajele de programare VS Pascal, VS Fortran și C.

Microprocesoare

Microprocesoare pe 8 biţi

Cipurile pe 8 biţi sînt încă elemente de bază în industria calculatoarelor, chiar dacă au fost înlocuite pe piaţă de cele pe 16 respectiv 32 de biţi. Microprocesoarele pe 8 biţi, cele mai cunoscute pe care le analizăm în cele ce urmează sînt: Intel 8080A, Intel 8085A, Z80, NSC800, MC6800, MC6809 şi MOS65202.

INTEL 8080A

Deşi Intel 8080 este deja o istorie antică într-o scară de timp comprimată a lumii microprocesoarelor, cipul este încă disponibil pentru utilizare pe scară largă avînd ca avantaje costul şi familiaritatea pentru programatori. Este bine cunoscut şi a fost în mod sigur testat. Experienţa pe Intel 8080A pregăteşte pe utilizatorii lui Z80, 8085, 8086 şi 8088.

Lista instrucţiunilor în ordine alfabetică. ADD x; **ADC** x; **ADI** (8); **ACI** (8); **ANI** (8); **ANA** x; **ANA** M; **CMP** x; **CMA**; **CPI** (8); **CALL** (adr); **CNZ** (adr); **CZ** (adr); **CNC** (adr); **CC** (adr); **CPO** (adr); **CPE** (adr); **CP** (adr); **CM** (adr); **DAD** B; **DAD** D; **DAD** SP; **DCR** X; **DCX** B; **DCX** D; **DCX** H; **DCX** SP; **DAA**; **DI**; **EI**; **HALT**; **INR** X; **INX** B; **INX** D; **INX** H; **INX** SP; **IN** (port); **JMP** (adr); **JNZ** (adr); **JZ** (adr); **JNC** (adr); **JC** (adr); **JPO** (adr); **JPE** (adr); **JP** (adr); **JM** (adr); **LXI** B,(16); **LXI** D,(16); **LXI** H,(16); **LXI** SP(16); **LDAX** B; **LDAX** D; **LHLD** (adr); **LDA** (adr); **MOV** X,X; **MVI** X,(n); **MVI** M,(8); **NOP**; **ORI** (8); **OUT** (port); **ORA** X; **PUSH** B; **PUSH** D; **PUSH** H; **PUSH PSW**; **POP** B; **POP** D; **POP** H; **POP PSW**; **RLC**; **RRC**; **RAL**; **RAR**; **RET**; **RNZ**; **RZ**; **RNC**; **RC**; **RPO**; **RPE**; **RP**; **RM**; **RST** n; **STAX** B; **STAX** C; **SHLD** (adr); **STA** (adr); **SUB** X; **SBB** X; **STC**; **SUI** (8); **SBI** (8); **SPHL**; **XCHG**; **XRA** X; **XRI** (8); **XTHL**.

Calculatoare cu microprocesor 8080. CUB (România), TPD (România), D.A.I. (Belgia) etc.

INTEL 8085 A

Microprocesorul 8085 este compatibil soft cu părintele său 8080 A deşi într-un circuit existent nu-l poate înlocui pe acesta. Oferă o viteză mai mare decît microprocesorul 8080 A. Integrează funcţiile generatorului ceas 8224 şi a controlerului sistem 8228, care iniţial au fost privite separat faţă de 8080 oferind pe această bază fabricanţilor de sisteme o unitate centrală de prelucrare integrată. Accesează 64 K memorie.

Calculatoare cu microprocesor Intel 8085 A. ALCYANE A 100E (Franţa), ALCYANE 6E (Franţa), P2U, P3U, P4U, P30, P40 (R.F.G.) etc.

ZILOG Z-80

Microprocesorul Zilog Z80 combină caracteristicile unice ale lui 8080 şi 6800. Este compatibil cu codul 8080. Foarte multe calculatoare sub CP/M, care este scris în 8080, sînt proiectate în jurul microprocesorului Z80. Uneori este numit a doua generaţie a lui 8080.

Lista instrucţiunilor în ordinea alfabetică. ADC A; **ADC** HL,rp; **ADD**; **AND**; **BIT**; **CALL** addr; **CALL** cond,addr; **CCF**; **CP**; **CPD**; **CPDR**; **CPI**; **CPIR**; **CPL**; **DAA**; **DEC**; **DI**; **DJNZ**; **EI**; **EX**; **EXX**; **HALT**; **IM** n; **IN**; **INC**; **IND**; **INDR**; **INI**; **INIR**; **JP** addr; **JP** cond, addr; **JR**; **JR** cond,addr; **LD** reg,(HL); **LD** A(addr); **LD** data; **LD** (HL),reg; **LD** (addr),A; **LD** A,I; **LDA**,R; **LD** dst,src; **LDA**,(BC) sau (DE); **LD** HL,(addr); **LD** I,A; **LD** R,A; **LD** reg,(xy+disp); **LD** rp,(addr); **LD** xy,(addr); **LD** (BC) sau (DE), A; **LD** (addr),HL; **LD** (xy+disp),reg; **LD** (addr),rp; **LD** (addr),xy; **LD** (HL),data; **LD** (xy+disp),data; **LD** SP,HL; **LD** SP,xy; **LDD**, **LDDR**; **LDI, LDIR**; **NEG**; **NOP**; **OR**; **OUT**; **OUTD**; **OTDR**; **OUTI**; **OTIR**; **POP**; **PUSH**; **RES**; **RET**; **RETI**; **RETN**; **RLA**; **RL**; **RLC**; **RLCA**; **RRA**; **RR**; **RRC**; **RRCA**; **RLD**; **RRD**; **RST**; **SBC**; **SCF**; **SET**; **SLA**; **SRA**; **SRL**; **SUB**; **XOR**.

Calculatoare cu microprocesor Z80 (Z80A). aMIC (România), PRAE (România), HC-85 (România), TIM S (România), JUPITER ACE (Anglia), SANYO PHC 25 (Japonia), SPECTRUM (Anglia), VICTOR λ HR (S.U.A.), SINCLAIR ZX-81 (Anglia), AVC 777 (Japonia), KAYPRO II (S.U.A.), MAI 10 (S.U.A.), SORD (Japonia), NEW BRAIN (Anglia), OSBORNE EXECUTIVE (S.U.A.), AMSTRAD (model PCW 8512), ABC 24–26 (Japonia), ADDX SUPER MICRO (Franţa),

OLYMPIA BOSS (R.F.G.), P2010, P2012, P2009 (R.F.G.), TIKI 100 (Norvegia), ICL PC (Anglia), IF800 (Japonia), ILDA-2 (Franța), LOGABAX LX 528 (Franța), M 243 (Japonia), TOSHIBA T100 (Japonia), TRS 80 II, TRS 80 III, TRS 80-12 (S.U.A.), XEROX 820 (S.U.A.), MERITUM I (R.P.P.), ROBOTRON 9001 (R.D.G.), ROBOTRON 1715 (R.D.G.), MERITUM II (R.P.P.), PRIMO (R.P.U.) etc.

NSC 800

Microprocesorul NSC 800 combină elemente caracteristice lui 8085 și Z80. Este un cip pe 8 biți apt de a lucra pe softul lui Z80 sau 8085.

Lista instrucțiunilor în ordine alfabetică. ADD; ADC; ADD A; ADC A; ADD A, m; ADC A, m; AND; AND n; AND m; BIT; CP; CP n; CP m; CPL; CCF; CPI; CPD; CPIR; CPDR; CALL nn; CALL cc,nn; DAA; DEC; DEC m; DI; DJNZ; EX DE,HL; EX AF,AF'; EXX; EX (SP); EI; HALT; INC; INC m; IN A,n; IN r,C; INI; IND; INIR; INDR; IM 0; IM 1; IM 2; JP; JR; JR cc; LD; LD ACC; LD SP,ss; LD (16 — bit binary n),rr; LDI; LDD; LDIR; LDDR; OR; OR n; OR m; OUT n,A; OUTC, r; OUTI; OUTD; OUTIR; OUTDR; POP; PUSH; RES; RET; RET CC; RETI; RETN; RST T; RLC r; RL r; RRC r; RR r; RLC m; RL m; RRC m; RR m; RLD; RRD; SUB; SBC; SUB n; SBC A,n; SUM m; SBC A,m; SLA m; SRA m; SCF; SET; SRL m; SLA r; SRA r; SRL r; XOR; XOR n; XOR m.

MOTOROLA MC 6800

Microprocesorul MC 6800 este orientat spre memorie. Are puține instrucțiuni ce se referă la registre prin aceasta deosebindu-se puternic de microprocesorul 8080.

Lista instrucțiunilor în ordine alfabetică. ADD A; ADD B; ABA; ADCA; ADCB; ANDA; ANDB; ASL; ASLB; ASR; ASRA; ASRB; BITA; BITB; BRA; BCC; BCS; BEQ; BGE; BGT; BHI; BLE; BLS; BLT; BMI; BNE; BVC; BVS; BPL; BSR; CLR; CLRA; CLRB; CMPA; CMPB; CBA; COM; COMA; COMB; CLC; CPX; CLI; CLV; DAA; DEC; DECA; DECB; DEX; DES; EORA; EORB; INC; INCA; INCB; INX; INS; JMP; LDAA; LDAB; LSR; LSRA; LSRB; LDX; LDS; NEG; NEGA; NEGB; NOP; ORAA; ORAB; PHSA; PHSB; PULA; PULB; ROL; ROLA; ROLB; ROR; RORB; RTI; RTS; STAA; STAB; SBA; SBCA; SBCB; STX; STS; SEI; SEV; SWI; TAB; TBA; TST; TXS; TSX; TAP; TPA; WAI.

MOTOROLA MC6809

Microprocesorul MC6809 este asemănător cu microprocesorul 6800, dar totuși un program scris în cod obiect 6800 nu poate fi executat pe microprocesorul MC6809. Motorola MC6809 atinge puterea de calcul pentru 16 biți. De asemenea, filozofia este orientată spre memorie. MC6809 este capabil la 1121 operații distincte (59 de instrucțiuni multiplicate cu 19 moduri de adresare).

Lista instrucțiunilor în ordine alfabetică. ABX; ADC; ADD; AND; ANDCC; ASL/LSL; ASR/LSR; BCC/BHS; BCS/BLO; BEQ; BGE; BGT; BHI; BIT; BLE; BLS; BLT; BMI; BNE; BPL; BRA; BRN; BSR; BVC; BVS; CLR; CMP; COM; CWAI; DAA; DEC; EOR; EXG; INC; JMP; JSR; LD; LEA; MUL; NEG; NOP; OR; ORCC; PSH; PUL; ROL; ROR; RTI; RTS; SBC; SEX; ST; SUB; SWI; SYNC; TFR; TST.

Calculatoare cu microprocesor MC6809. THOMSON TO 7 (Franța) etc.

MOS 6502

Microprocesorul MOS 6502 este o versiune eficientă a microprocesorului Motorola MC6800.

Lista instrucțiunilor în ordine alfabetică. ADC; AND; ASL; BCC; BCS; BEQ; BIT; BMI; BNE; BPL; BRK; BVC; BVS; CLC; CLD; CLI; CLV; CMP; CPX; CPY; DEC; DEX; DEY; EOR; INC; INX; INY; JMP; JSR; LDA; LDX; LDY; LSR; NOP; ORA; PHA; PLA; PLP; ROL; ROR; RTI; RTS; SBC; SEC; SED; SEI; STA; STX; STY; TAX; TAY; TSX; TXA; TXS; TYA.

Calculatoare cu microprocesor MOS 6502. HHC PANASONIC (Japonia), ATARI 400 (S.U.A.), ATARI 800 (S.U.A.), BBC ACCORN (Anglia), COMMODORE VIC 20 (S.U.A.), APPLE II, APPLE III (S.U.A.), MULTITECH MPF II (Taivan), PRAVETZ (R.P.B.) etc.

Microprocesoare pe 8/16 biţi

Între microprocesoarele pe 8 biţi şi cele (adevărate) pe 16 biţi există o grupă hibridă care partajează anumite limitări ale celor din prima grupă cu avantajele celor din a doua grupă.

TEXAS INSTRUMENTS TMS9900

Microprocesorul TMS9900 utilizează o arhitectură pe 16 biţi fiind orientat spre memorie şi putînd adresa doar 64 K.

Lista instrucţiunilor în ordine alfabetică. A; AB; AI; ANDI; ABS; B; BL; C; CB; CI; BLWP; COC; CLR; CZC; DIV; DEC; INC; INV; JMP; JMP (cond); **LDCR; LI** (reg); **MOV; MOVB; MPY; NEG; ORI; RTWP; SWPB; S; SB; SOC; SOCB; SZC; SZB; SLA; SRA; SRL; SRC; STWP; STCR; SBO; SBZ; SETTO; TB; XOR; XOP; X** src.

Calculatoare cu microprocesor TMS9900. TI 99/40 (S.U.A.),

INTEL 8088

Datorită faptului că magistrala externă are 8 biţi microprocesorul Intel 8088 poate utiliza cipuri suport de 8 biţi. Cipul are o magistrală de adresare de 8 biţi permiţînd un spaţiu de adresare de 1 MO. Intel 8088 are două unităţi de prelucrare separate: unitatea de execuţie (execută toate instrucţiunile) şi unitatea interfaţă magistrală (încarcă instrucţiunile şi transportă datele între magistrala internă de 16 biţi şi cea externă de 8 biţi). Se creează astfel posibilitatea ca microprocesorul 8088 şi vărul lui de 16 biţi – 8086 să efectueze funcţii de calculare internă în timp ce se încarcă şi se expediază datele.

Lista instrucţiunilor în ordine alfabetică. ADD; ADDB; ADDI; ADDBI; ADC; ADCB; ADCI; ADCBI; AND; ANDB; ANDI; ANDIB; AAA; AAM; AAD; CMC; CBW; CWD; CMPC; CMPW; CLD; CMP; CMPB; CMPI; CMPBI; CALL; CALLI; CALLL; CLI; CLC; DIV; DIVB; DEC; DECB; DAA;DAS; ESC; HLT; IN; INB; IMUL; IMULB; IDIV; IDIVB; INC; INCB; INT; INTO; IRET; JMP; JMPS; JMPI; JMPL; JE; JZ; JNE; JNZ; JS; JNS; JP; JNP; JPE; JPO; JL; JNGE; JNL; JGE; JLE; JNG; JNLE; JG; JB; JNAE; JNB; JAE; JBE; JNA; JNBE; JCXZ; LEA; LDS; LES; LODC; LODW; LOOP; LOOPZ; LOOPNZ; LOOPE; LOOPNE; LOCK; LAHF; MOV; MOVB; MOVI; MOVBI; MUL; MULB; MOVC; MOVW; NOT; NEG; NEGB; NOTB; NOP; OUT; OUTB; OR; ORB; ORI; ORIB; PUSH; POP; PUSHF; POPF; ROL; ROLB; ROR; RORBI; RCL; RCLB; RCR; RCCB; REP; RET; RET n; RETS; RETS n; SUB; SUBB; SUBI; SUBBI; SBB; SBBB; SBBI n; SBBBI; SHL; SHLB; SHR; SHRB; SAL; SALB; SAR; SARB; SCAC; SCAW; STC; STOC; STD; STOW; STI; SAHF; TEST; TESTB; TESTI; TESTBI; XCHG; XCHGB; XLAT; XOR; XORB; XORI; XORIB; WAIT.

Calculatoare cu microprocesor Intel 8088, XT (R.F.G.), IBM PC jr (S.U.A.), COMPAQ (model Dual/Plus), AMSTRAD (model PC 1512), TANDY (model 1000 EX), TANDY (model 1000 SX), ZENITH (model Z148 PC), ZENITH (model 7158 PC), PC IBM XT, DUET 16 (Japonia), CANON AS 100 (Japonia), AXEL AX 20 (Franţa), CORONA PC (S.U.A.), HITACHI 16000 (Japonia), VICTOR S1 (S.U.A.), VECTOR 4 (S.U.A.), INTELEXT (R.P.B.), SMEP PP06 (R.S.C.), PROPER 16 B (R.P.U.), ELECTRONICA MC 1201 (U.R.S.S.) etc.

Microprocesoare pe 16 biţi

Microprocesoarele pe 16 biţi pe care le descriem în cele ce urmează sînt: Intel 8086, Zilog Z8000, Zilog Z8002 şi Motorola 68000.

INTEL 8086

Microprocesorul Intel 8086 este o versiune pe 16 biţi a microprocesorului 8088. Programele scrise pentru microprocesoarele 8088 şi 8086 sînt reciproc compatibile. Intel 8086 are aceeaşi arhitectură, mod de adresare şi set de instrucţiuni ca 8088 cu excepţia unităţii de interfaţă magistrală (8086 comunică pe magistrală de 16 biţi).

Calculatoare cu microprocesor Intel 8086. FELIX PC (România), EPSON (model PC+), OLIVETTI (model M24), OLIVETTI (model M24 SP), VICTOR (model VPC 2), APRI-

COT (Anglia), MICRON 20, MICRON 200, MICRON 2200 (Norvegia), A.I. M16 (Japonia), ALTOS 586 (S.U.A.), BFM 186 (Franța), SORD M343 MARK X (Japonia), WANG PC 001 (S.U.A.), TULIP COMPUDATA (Olanda) etc.

ZILOG Z8000

Microprocesorul Zilog Z8000 are de fapt două versiuni: Z8001 ce adresează 8 Mo și Z8002 limitată la 64 Ko. Codul de operație este pe 16 biți.

Lista instrucțiunilor în ordine alfabetică. ADD; ADDB; ADDL; AND; ANDB; ADC; ADCB; ANDL; BIT; BITB; CLR; CLRB; COM; COMB; CP; CPB; CPL; CPD; CPDB; CPDR; CPDBR; CPI; CPIB; CPIR; CPIBR; CPSD; CPSDB; CPSDR; CPSDRB; CPSI; CPSIB; CPSIR; CPSIRB; CLR; CLRB; COM; COMB; CALL; CALR; COMFLG; DEC; DECB; DIV; DIVL; DJNZ; DBJNZ; DAB; DI; EX; EXB; EXTS; EI; EXTSB; EXTSL; HALT; IN; INB; IND; INDB; INDR; INDRB; INI; INIB; INIR; INIRB; INC; INCB; IRET; JP; JR; LD; LDB; LDL; LDA; LDAR; LDR; LDRB; LDK; LDD; LDDB; LDDR; LDDRB; LDI; LDIB; LDIR; LDIRB; LDM; LDCTL'; LDCTL; LDCTLB; LDCTL Refresh; **MULT; MULTL; MBIT; MREQ; MRES; MSET; NEG; NEGB; NOP; OTDR; OTDRB; OTIR; OTIRB; OUT; OUTB; OUTD; OUTDB; OUTI; OUTIB; OR; ORB; POP; POPL; PUSH; PUSHL; RES; RESB; RL; RLB; RET; RLC; RLCB; RR; RDLB; RRDB; RRB; RRC; RRCB; RESFLG; SIN; SINB; SIND; SINDB; SINDR; SINDRB; SINI; SINIB; SINIR; SINIRB; SOTDR; SOTDRB; SOTIR; SOTIRB; SOUT; SOUTB; SOUTD; SC; SOUTDB; SOUTI; SOUTIB; SUB; SUBB; SUBL; SBC; SBCB; SET; SRAL; SDA; SDAB; SDAL; SETFLG; SDL; SDLB; SDLL; SLA; SLAB; SLAL; SRL; SRLB; SRLL; SLL; SLLB; SLLL; SRA; TEST; TESTB; TESTL; TSET; TSETB; TRDB; TRDRB; TRIB; TRIRB; TRTDB; TRTDRB; TRTIB; TRTIRB; TSET; TSETB; TCC; TCCB; XOR; XORB.**

ZILOG Z8002

Microprocesorul Z8002 este prevăzut cu 40 de pini în locul celor 48 de pini ai lui Z8001. Z8002 are o caracteristică soft care pune cipul într-un mod supervizor. Motivul este simplu deoarece Z8000 cuprinde o serie de operații I/O, instrucțiuni de control hard și instrucțiuni de manipulare întreruperi care dacă sînt utilizate defectuos de către un începător pot serios corupe operarea cipului într-un mediu utilizator. Crearea unui mediu privilegiat de utilizare a fost prevăzută pentru a preveni acest fenomen, pentru a trece comenzile de control I/O și hard critice către o stare supervizor, mod care poate fi invocat numai de către sistemul de întreruperi externe.

Microprocesoare pe 16/32 biți

MOTOROLA 68000

Calculatoarele Apple Lisa și Macintosh care utilizează pe scară largă grafica electronică au la bază microprocesorul Motorola 68000. Tehnicile de gestiune a memoriei sofisticate precum și posibilitatea manipulării datelor în segmente de 8, 16 și 32 de biți au făcut ca Motorola 68000 să poată fi comparat cu calculatoarele mari. Riguros vorbind, Motorola 68000 este un hibrid 16/32 biți avînd o adresare a memoriei pe 32 de biți. Microprocesorul 68000 are două moduri de operare: supervizor și utilizator. În primul mod de operare (supervizor) se execută numai anumite comenzi privilegiate ce sînt prioritare față de instrucțiunile în mod utilizator.

Lista instrucțiunilor în ordine alfabetică. ABCD; ADD.B; ADD.W; ADD.L; ADDX.B; ADDX.W; ADDX.L; AND.B; AND.W; AND.L; ADDQ.B; ADDQ.W;ADDQ.L; ADDX.B; ADDX.W; ADDX.L; ASL; ASL.B; ASL.W; ASL.L; ASR; ASR.B; ASR.W; ASR.L; BSR; Bcc; BTST; BSET; BCLR; BCHG; CLR.B; CLR.W; CLR.L; CMP.B; CMP.W; CMP.L; CMPM.B; CMPM.W; CMPM.L; CLR.B; CLR.W; CLR.L; CHK; DIVS; DIVU; DBcc; EOR.B; EOR.W; EOR.L; EXG; EXT.W; EXT.L; JMP; JSR; LEA; LSL; LSL.B; LSL.W; LSL.L; LSR; LSR.B; LSR.W; LSR.L; LINK; MOVE.B; MOVE.W; MOVE.I; MOVEM.W; MOVEM.L; MOVEP.L; MULS; MULU; MOVEQ; MOVE.B; MOVE.W; MOVE.L; MOVE; NBCD; NEG.B; NEG.W; NEG.L; NEGX.B; NEGX.W; NEGX.L; NOT.B; NOT.W; NOT.L; NBCD; NOP; OR.B; OR.W; OR.L; PEA; RTS; RTR; ROL; ROL.B; ROL.W; ROL.L; ROR; ROR.B; ROR.W; ROR.L; ROXL; ROXL.B; ROXL.W; ROXL.L; ROXR; ROXR.B; ROXR.W; ROXR.L; RTE; RESET; SBCD; SCC; SUB.B; SUB.W; SUB.L; SUBX.B; SUBX.W; SUBX.L; SUBQ.B; SUBQ.W; STOP; Scc; SWAP; TAS; TST.B; TST.W; TST.L; TRAP; UWLK.

Calculatoare cu microprocesor Motorola 68000. APPLE (model Macintosh Plus), APPLE (model Macintosh 512 K/800), ATARI (model 1040 ST), COMMODORE (model Amiga), MICROOMEGA (Franţa), OLYMPIA PEOPLE (R.F.G.), CROMENCO (S.U.A.) etc.

□ 19 limbaje de programare mai mult sau mai puţin cunoscute. Prezentare în ordine alfabetică, cuvinte cheie

Ada

Ada este mai mult decît un limbaj de programare, este un sistem complet pentru elaborarea produselor soft şi utilizează construcţii standard sub formă de blocuri în alcătuirea programelor. Ada reflectă caracteristicile importante necesare elaborării unor produse soft de calitate şi anume: simplitate şi completitudine, siguranţă în realizarea programelor, corectitudine, întreţinere uşoară, portabilitate, programare în timp real şi tratare a erorilor. Ada reflectă, de asemenea, tendinţele de utilizare a calculatoarelor mari şi a arhitecturilor multiprocesor, fiind totodată folosit în programarea concurentă.

Numele de Ada provine de la Augusta Ada Byron, contesă de Lovelace, fiica poetului Byron. Ada a programat maşina analitică a lui Babbage pentru a calcula numerele lui Bernoulli, fiind considerată în istoria calculatoarelor ca prima programatoare.

Programele scrise în limbajul Ada sînt de obicei elaborate pentru calculatoare mari, superminicalculatoare de 32 de biţi cum ar fi DEC VAX.

Cuvinte rezervate. ACCEPT, ACCESS TYPE, ACTUAL PARAMETER, AGGREGATE, ALLOCATOR, ANCESTOR COMPILATION UNIT, ARRAY TYPE, ASSIGNMENT, ASSOCIATION, ATTRIBUTE, BLOCK STATEMENT, BODY, CHARACTER, COLLECTION, COMPILATION, COMPILATION UNIT, COMPLETE PROGRAM, COMPONENT, COMPUTER PROGRAM COMPONENT, COMPUTER PROGRAM CONFIGURATION ITEM, CONSTANT, CONSTRAINT, CONTEXT CLAUSE, DECLARATION, DECLARATIVE PART, DENOTE, DEPENDENCE RELATION, DERIVED TYPE, DESIGNATE, DIRECTORY MODE, DISCRETE TYPE, DISCRIMINANT, ENTITY, ENTRY, ENUMERATING TYPE, EXCEPTION, EXPANDED NAME, EXPRESSION, FILE NODE, FORMAL PARAMETER, FOSTER PARENT, FUNCTION, GENERIC UNIT, HANDLER, HOST, IDENTIFIER, INCOMPLETE PROGRAM, INDEX, INDEX CONSTRAINT, INDEXED COMPONENT, INTEROPERABILITY, INSTANCE, INTEGER TYPE, LEXICAL UNIT, LIBRARY SUBPROGRAM BODY, LIMITED TYPE, LITERAL, LOCATOR, LOGGED PROGRAM, LOGGED SEQUENCE, MAIN PROGRAM, MODE, MODE ANCESTORS, MODEL NUMBER, MODULE, NAME, NAMED ASSOCIATION, OBJECT, OPERATION, OPERATOR, ORIGINAL CONTAINER, OVERLOADING, PACKAGE, PARAMETER, PARENT TYPE, POSITIONAL ASSOCIATION, PRAGMA, PARENT NODE, PREFIX, PRIVATE PART, PRIVATE TYPE, PROCEDURE CALL, PROGRAM, PROGRAM LIBRARY, QUALIFIED EXPRESSION, RANGE, RANGE CONSTRAINT, REAL TYPE, RECORD TYPE, REHOSTABILITY, RENDEZVOUS, REPRESENTATION CLAUSE, RETARGETABILITY, REUSABILITY, REVISION, REVISION SET, SCALAR TYPE, SCOPE, SELECTED COMPONENT, SELECTOR, SERIAL NUMBER, SIMPLE NAME, SLICE, SNAPSHOT CONTAINER, STATEMENT, STRING, SUBCOMPONENT, SUBPROGRAM, SUBTYPE, SUBUNIT, TARGET, TASK, TRANSPORTABILITY, TRUE PARENT, TYPE, USABLE CONTAINER, USE CLAUSE, VARIABLE, VARIANT, VARIATION SET, VISIBILITY, VISIBLE PART, WITH CLAUSE.

Algol

Algol-ul este unul din cele mai vechi şi cele mai puţin utilizate limbaje de programare. Este un limbaj algoritmic cu destinaţie generală pentru rezolvarea informatică a problemelor. A fost elaborat în 1958 cu doi ani mai tîrziu ajungîndu-se la varianta Algol 60.

Algol 60 admite multe concepte matematice avansate fiind însă lipsit de facilităţi de manipulare a şirurilor. Versiunea curentă standard – Algol 68 corectează multe din deficienţele versiunii anterioare.

O versiune limitată a lui Algol 68 este disponibilă sub sistemul de operare CP/M.

Cuvinte rezervate (Algol sub CP/M). **BEGIN, BIT, BOOL, BY, BYTE, CASE, CHAR, CO, DIVAB, DO, ELSE, END, ESAC, FALSE, FOR, FORMAT, FROM, GOTO, HEAP, IF, IN, INT, LABEL, LOC, LONG, MINUSAB, MULTAB, OD, OF, OUT, PLUSAB, PRINT, PROC, READ, REAL, REF, SHORT, STRUCT, THEN, TRUE, WHILE.**

APL

Limbajul APL a fost elaborat de Kenneth Iverson în anul 1962. Este considerat un limbaj algebric cu destinație generală fiind ideal pentru rezolvarea problemelor științifice și inginerești. O versiune modificată a limbajului – APL/360 a fost implementată pe calculatorul IBM 360 pentru a rezolva probleme din domeniul analizei Fourier, metoda celor mai mici pătrate, integrare numerică, probleme de valori proprii, ecuații Laplace (prin metode numerice).

Îndată ce utilizatorul furnizează un enunț singular acesta este și executat. O caracteristică importantă a limbajului este capacitatea sa de a manipula vectori și matrici. Se poate lucra în mod imediat dar și în mod programat.

C

Limbajul C a fost proiectat de către Dennis M. Ritchie în anul 1972 și a fost descris în "C Reference Manual", publicat în ianuarie 1974 de către laboratoarele Bell. A devenit foarte popular pe o mare varietate de microprocesoare deoarece modul de proiectare a lui **C** permite o mare portabilitate. Caracteristica cea mai importantă a limbajului este aceea că permite programarea într-un cod care să poată fi utilizat ușor pe cît mai multe procesoare posibile. Este un limbaj structurat, bazat cu precădere pe funcții. Începînd din 1983 și 1984, versiuni ale compilatorului **C** au fost anunțate de Micro Soft și Digital Research. Compilatoarele de **C** sînt disponibile actualmente pe cele mai multe microprocesoare. Scopul său inițial a fost acela de a se constitui ca un instrument în asistarea elaborării sistemului de operare UNIX pe minicalculatorul DEC PDP-11.

Cuvinte rezervate. ALLOC, AND&&, a.out, array declaration, ATOF, ATOI, binary, bitcount, break, %c, CASE, cfree, char, close, copy, create, %d, dv, extern, exit, %f, fclose, float, fopen, for, fprintf, fputs, free, fscanf, getbits, getc, getch, getchar, getint, getop, goto, if, if-else, index, install, int, itoa, %ld, long, lookup, lower, main, month, morecore, null, numcmp, %o, open, printf, putc, putchar, return, reverse, %s, scanf, sprintf, squeeze, sscanf, static, strcat, strcmp, strcpy, strlen, strsave, switch, typedef, while, %x.

COBOL

Limbajul COBOL (Common Business Oriented Language) a fost oficial definit în 1968 și din nou în 1974 de către ANSI (American National Standards Institute). În prezent, 60 pînă la 75% din codul noilor aplicații se scriu în COBOL. Viitorul COBOL-ului oferă o mare promisiune pentru rețelele de microcalculatoare. Precizia COBOL-ului cît și capacitatea sa de-a manipula numere mari îl fac încă un limbaj ales pentru gestiune iar disponibilitatea sa pe calculatoare de 8 biți îi conferă o mare portabilitate.

Cuvinte rezervate. ACCEPT, ADD, ALL, ALTER, AND, ASSIGN, AT END, AUTHOR, BLANK, CLOSE, COMMA, COMPUTATIONAL (COMP), COMPUTATIONAL-3 (COMP-3), COMPUTE, CONFIGURATION SECTION, CORRESPONDING (CORR), DATA DIVISION, DATE-COMPILED, DATE-WRITTEN, DATE, DELIMITED, DEPENDING, DISPLAY, DIVIDE, ELSE, ENVIRONMENT DIVISION, EXIT, FD, FILE CONTROL, FILLER, FROM, GIVING, GOTO, GREATER, GROUP INDICATE, HEADING, HIGH-VALUES, IDENTIFICATION DIVISION, IF, INDEX, INDICATE, INPUT-OUTPUT SECTION, INSPECT, INSTALLATION, INTO, JUSTIFIED (JUST), LABEL, LOW-VALUES, MOVE, MULTIPLY, NEXT SENTENCE, NUMERIC LITERAL, OBJECT-COMPUTER, OCCURS, OMITTED, OPEN, OR, ORGANIZATION, PAGE-COUNTER, PERFORM, PICTURE (PIC), PROCEDURE DIVISION, PROGRAM-ID, READ, RECORD, REDEFINES, REMAINDER, RENAMES, RETURN, ROUNDED, SEARCH ALL, SEARCH, SECURITY, SELECT, SET, SIGN, SIZE-ERROR, SOURCE-COMPUTER, SPACES, STANDARD, STOP, STRING, SUBTRACT, SYNCHRONIZED (SYNC), UNSTRING, UNTIL, USAGE, VALUE, VARYING, WHEN, WORKING-STORAGE SECTION, WRITE, ZERO.

COMAL

COMAL-80 este un limbaj de nivel înalt, elaborat în Danemarca. Limbajul combină caracteristicile BASIC-ului cu ale PASCAL-ului. Autorul limbajului este Borge Christensen. Prima implementare a COMAL-ului a fost realizată pe minicalculatorul Data General

Nova 1200. Din anul 1978 a devenit foarte popular în școlile daneze. În anul 1979 a fost definită o nouă versiune a limbajului cunoscută sub denumirea de COMAL-80.

Cuvinte rezervate. ABS, AND, ATN, CASE, CAT, CHAIN, CHR$, CLOSED, CON, COS, DATA, DIM, ELIF, ELSE, ENDCASE, ENDIF, ENDPROC, EXEC, EXP, FOR, GOTO, IF, IN, INPUT, INT, LEN, LOG, MOD, NOT, OPEN, OR, ORD, OTHERWISE, PRINT, PROC, RANDOMIZE, READ, REPEAT, RND, SELECT OUTPUT, SIN, STEP, SQR, TAB, TAN, UNTIL, WHEN, WHILE, ZONE.

Forth

Forth este mai mult decît un limbaj de programare, este un mediu de programare extensibil atît în interior cît și în exterior. Extensia externă se realizează pentru a adăuga un nivel de limbaj de aplicație și a utiliza apoi acest nou limbaj de aplicație pentru a adăuga în continuare alte niveluri (de aplicație). Extensia internă se realizează prin modificarea structurii posibilităților și caracteristicilor limbajului (Forth) — proces denumit metacompilare, un proces prin care Forth-ul recompilează o nouă versiune de Forth. Forth-ul a fost elaborat în anul 1971, avîndu-l ca autor pe Charles H. Moore. În 1978 s-au realizat versiunile pentru PDP 11 și microcalculatoare. Din 1988 este disponibil pe calculatoarele de 32 de biți. Forth este concomitent: un sistem de operare, un editor, un asamblor, un interpretor, un compilator, un limbaj de nivel înalt, un set de instrumente de elaborare, o mașină de tip stivă.

Cuvinte rezervate (Forth-83). ABS, ALLOT, AND, BASE, BEGIN, BLK, BLOCK, BUFFER, C!, C@, CMOVE, COMPILE, CONSTANT, CONVERT, COUNT, CR, CREATE, D+, D<, DECIMAL, DEFINITIONS, DEPTH, DNEGATE, DO, DOES>, DROP, DUP, ELSE, EMIT, EXECUTE, EXIT, EXPECT, FILL, FIND, FLUSH, FORGET, FORTH, FORTH-83, HERE, HOLD, I, IF, IMMEDIATE, J, KEY, LEAVE, LITERAL, LOAD, LOOP, MAX, MIN, MOD, NEGATE, NOT, OR, OVER, PAD, PICK, QUIT, R>, R@, REPEAT, ROLL, ROT, SAVE-BUFFERS, SIGN, SPACE, SPACES, SPAN, STATE, SWAP, THEN, TIB, TYPE, U., U<, UM*, UM/MOD, UNTIL, UPDATE, VARIABLE, VOCABULARY, WHILE, WORD, XOR.

PC Forth

Limbajul PC Forth a fost scris special pentru PC-uri și reprezintă un super set al standardului Forth-83 pentru a putea fi rulat pe calculatoarele compatibile 8086, 8088.

Cuvinte rezervate (extensii ale lui Forth pentru calculatoare IBM PC și PC-uri compatibile). ABS, ASSEMBLER, AND, addr, ABORT, ALLOT, ASCII, ALSO, AGAIN, ATTRIBUTE, b, BINARY, BLANK, BLOCK, BUFFER, BEGIN, B/BUFF, BELL, BL, BLK/SIDE, BS, BACKGROUND, BASE, BLK, BLOCK-R/W, char, CONVERT, C!, C!L, C@, CRESET, CSET, CTOGGLE, CMOVE, CMOVE>, CMOVEL, COMPARE, COUNT, CR, C, COMPILE, COMPILE-ONLY, CONTROL, CASE:, CREATE, CODE, CONSTANT, COLD, CAPS, COMM, COMM#, CRT-MODE, CONTEXT, CURRENT, CLEAR SCREEN, CRT-MODE!, CRT-MODE?, CRT-TTY, d, +d, ?DUP, 2DROP, 2DUP, 3DUP, DEPTH, DROP, DUP, DEPTH, D2/, D+, D–, DABS, DMAX, DNEGATE, DMIN, DO=, D<, D=, D>, DU<, DECIMAL, DIGIT, D., D.R, DISCARD, DIR, DUMP, DO, ?DO, DEFINITIONS, DLITERAL, DEFER, DOES>, DISK, DISK-ERROR, DP, DPL, DISK-RESET, ERASE, EXPECT, EMIT, EMPTY-BUFFERS, ESTABLISH, ELSE, EXIT, EDITOR, .ENDIF, EXECUTE, END-CODE, END?, ENTRY, flag, FILL, FLUSH, FIND, FORGET, FORTH-83, FALSE, FIRST, FOREGROUND, FENCE, false, GO, HERE, HEX, HLD, HOLD, H., INDEX, I, IF, .IF, .INDF, .IFNDF, IS, INTERPRET, J, QUIT, QUERY KEY, ?KEY, K, LINELOAD, LOAD, LIST, +LOOP, ?LEAVE, LEAVE, LOOP, LATEST, LITERAL, LIMIT, LINK, M/MOD, MAX, MOVE, MIN, MOD, MOUNT, MAX-BUFS, mode, n, NEGATE, NOT, NUMBER, NUMBER?, OVER, OR, ORDER, ON, OFF, OCTAL, OFFSET, PICK, PAD, PERFORM, PRINTER, PRINTER#, PRINTING, P!, P@, PC!, PC@, PRINTER-INIT, PRINTER-OUT, PRINTER-STATUS?, R@, R>, ROLL, ROT, RP!, RPO, RP@, REPEAT, RE, RECURSE, REVEAL, R#, RPO, sys, SP!, SPO, SP@, S>D, SM/MOD, #S, SIGN, SPACE, SPACES, SAVE-BUFFERS, SHOW, SPO, SCR, STANDARD, span, SPAN, STATE, SET-VOLUME, SERIAL-IN, SERIAL-INIT, SERIAL-OUT, SERIAL-STATUS, SWAP, TIB, TYPE, TRIAD, THEN, TRUE, TIBO, TOS, u, ud, UM/MOD, U<, U>, U., U.R, UD., UD.R, UPDATE, USER, VIEW, VOCS, VARIABLE, VOCABULARY, #VOCS, VOLUME, VOC-LINK, wdun, WARM, WORD, WARNING, XOR.

Fortran

Fortran-ul (Formula Translating) a fost primul limbaj de nivel înalt care a fost mult utilizat în prelucrarea datelor. Este cel mai vechi limbaj evoluat care este încă uti-

lizat. Standardizarea limbajului a început în anul 1966, a continuat în 1978 iar în anul 1979 s-a elaborat FORTRAN 77. Fortran-ul este încă folosit deoarece reprezintă un instrument puternic în proiectarea aplicaţiilor ştiinţifice şi de gestiune. Rămîne de văzut dacă Fortran-ul se va bucura ca o renaştere ca limbaj de programare avînd în vedere apariţia noilor calculatoare pe 16 şi 32 biţi şi softul de emulare introdus de IBM şi AT&T.

Cuvinte rezervate (Fortran 77). ACCEPT, ASSIGN, BACKSPACE, CALL, CHARACTER, CLOSE, COMMON, COMPLEX, CONTINUE, DATA, DIMENSION, DO, DOUBLE PRECISION, END, ENDFILE, ENDWHILE, ENTRY, EQUIVALENCE, FORMAT, FUNCTION, GOTO, IF-THEN, IF-THEN-ELSE, IMPLICIT, INTEGER, LOGICAL, OPEN, PARAMETER, PAUSE, PRINT, READ, REAL, RETURN, REWIND, STOP, SUBROUTINE, TYPE, WHILE DO, WRITE.

LISP

LISP este unul dintre cele mai vechi limbaje de programare încă utilizate, al doilea după FORTRAN. In ciuda . . . bătrîneţii sale, LISP-ul este surprinzător de modern deoarece este capabil să încorporeze noi facilităţi.

LISP este un acronim de la List Processing sau List Programming şi-l are ca autor pe John Mc Carthy. Este prin excelenţă limbajul programării pentru inteligenţa artificială. S-au realizat chiar şi maşini LISP în care sistemul de operare, interpretoarele, compilatoarele, editoarele şi alte utilitare sînt scrise exclusiv în LISP. Se cunosc mai multe dialecte LISP: MACLISP, INTERLISP, FRANZ LISP, UCILISP şi ZETALISP.

Cuvinte rezervate (LISP sub MS-DOS). APPEND, APPLY, ARRAY, ASSOC, ATOM, BOUNDP, CAR, CATCH, CDR, CLOSE, CONCAT, COND, CONS, DEFMACRO, DEFSTRUCT, DEFUN, DO, EQL, EQUAL, EVAL, FUNCALL, GENSYM, GET, GETD, GO, IF, LAMBDA, LAST, LENGTH, LET, LIST, LISTP, MAP, MAPCAR, MEMBER, MINUSP, NCONC, NTH, NULL, NUMBERP, PLUSP, PRIN1, PRINT, PROG, PROGNA, PROG1, PROPS, PUT, QUOTE, READ, REMPROP, REST, REVERSE, RPLACA, RPLACD, RPLACP, SET, SETF, SETQ, SOME, STORE, STRINGP, SUBSET, SUBST, SUBSTRING, TERPRI, THROW.

Logo

Logo este un limbaj de prelucrare a listelor de nivel foarte înalt. Este o versiune specială a lui LISP. A fost elaborat la MIT (Massachusetts Institute of Technology) în 1960 pentru a fi un limbaj educaţional.

Cuvinte cheie (standardul MIT implementat pe APPLE II). ALLOF, ANYOF, BACK (BK), BACKGROUND (BG), BUTFIRST (BF), BUTLAST (BL), CATALOG, CLEARSCREEN (CS), DRAW, EDIT (ED), END, ERASE (ER), ERASEFILE, ERASEPICT, FIRST, FORWARD (FD), FULLSCREEN, HEADING, HIDETURTLE, HOME, IF, IFFALSE (IFF), IFTRUE (IFT), LAST, LEFT (LT), MAKE, NODRAW, NOT, NOTRACE, NOWRAP, OUTDEV, OUTPUT (OP), PADDLE 0, PADDLE 1, PADDLEBUTTON 1, PENCOLOR (PC), PENUP (PU), PENDOWN (PD), PRINT (PR), PRINT 1, PRINTOUT (PO), PRINTOUT TITLES (POTS), RANDOM, RANDOMIZE, RC?, READ, READCHARACTER (RC), READPICT, REMAINDER, REPEAT, REQUEST (RQ), RIGHT (RT), SAVE, SAVEPICT, SENTENCE (SE), SETHEADING (SETH), SETX, SETXY, SETY, SHOW-TURTLE (ST), SPLITSCREEN, SQRT, STOP, TEST, TEXTSCREEN, THING, TO, TOPLEVEL, TRACE, TURTLESTATE (TS), WRAP, XCOR, YCOR.

Modula-2

Niklaus Wirth a proiectat Modula în anul 1977 pentru scopuri speciale de multiprogramare. Modula-2 s-a "născut" în anul 1980 pentru a înlătura neajunsurile limbajului Pascal. Modula-2 moşteneşte sintaxa de bază a Pascal-ului şi tipurile de date din Pascal, de asemenea compilarea separată a modulelor de program, posibilitatea de multiprogramare etc.

Cuvinte rezervate. ABS, AND, ARRAY, BEGIN, BITSET, BOOLEAN, BY, CAP, CARDINAL, CASE, CHAR, CHR, CONST, DEC, DEFINITION, DISPOSE, DIV, DO, ELSE, ELSIF, END, EXCL, EXIT, EXPORT, FALSE, FLOAT, FOR, FROM, HALT, HIGH, IF, IMPLEMENTATION, IMPORT, IN, INC, INCL, INTEGER, LOOP, MOD, MODULE, NEW, NIL, NOT, ODD, OF, OR, ORD, POINTER, PROC, PROCEDURE, QUALIFIED, REAL, RECORD, REPEAT, RETURN, SET, THEN, TO, TRUE, TRUNC, TYPE, UNTIL, VAL, VAR, WHILE, WITH.

Pascal

Pascal-ul este un limbaj puternic structurat și este organizat în paragrafe sau blocuri de cod. Fiecare bloc conține o cantitate specială de informații sau set de instrucțiuni pentru a-l face autonom într-o secvență de program mai mare.

Aceste blocuri cărora li se atribuie nume în interiorul programelor sînt mult mai ușor de depanat și de identificat decît limbajele nestructurate gen BASIC care definesc liniile numai prin numere. Limbajul Pascal a fost elaborat în anul 1970 de către Niklaus Wirth în scopul predării disciplinei de programare a calculatoarelor ca pe o disciplină sistematică, puternic organizată și structurată. Proiectarea limbajului Pascal a încurajat codificarea structurată ca și o reconsiderare îngrijită și o analiză temeinică a tipurilor de date. Natura puternic structurată a Pascal-ului cît și simplitatea sa relativă au permis limbajului să poată fi implementat pe o mare varietate de calculatoare și deoarece este proiectat pe blocuri, programele pot fi ușor modificate din diverse medii ale sistemului. Odată cu apariția microcalculatoarelor Pascal-ul a fost privit ca pe un limbaj sofisticat ce poate fi complet implementat pe microcalculatoare puternice ca și pe calculatoare mari. Astfel, un program scris pentru un calculator IBM poate fi de asemenea implementat și pe Apple după numai cîteva ajustări privind dimensiunile de memorie. Se cunosc mai multe versiuni ale limbajului: UCSD standard, Apple Pascal, KMMM Pascal, Baby Pascal, Turbo Pascal ș.a.

Cuvinte rezervate (Turbo Pascal, KMMM Pascal, Baby Pascal pentru Commodore PET, Super PET, Commodore 64). ABS, AND, ARCTAN, ARRAY, BEGIN, CASE, CHAR, CHR, CONST, COS, DISPOSE, DIV, DOWNTO, ELSE, END, FILE, FOR, FUNCTION, GET, GOTO, IF, IN, INPUT, INTEGER, LN, MAXINT, MOD, NEW, NIL, NOT, ODD, OF, OR, ORD, OUTPUT, PACK, PACKED, PAGE, PRED, READ, PROCEDURE, REPEAT, SUCC, TYPE, WHILE, WRITE.

Pilot

Pilot (Programmed Inquiry, Learning Or Teaching) a fost creat de John A. Starkweather la Universitatea din California în 1968, ca un limbaj special pentru instruirea asistată de calculator. A fost creat ca un limbaj pentru studenți, profesori și alți non-programatori ce-l pot utiliza pentru elaborarea materialelor de instruire și administrarea testării studenților. Este un limbaj simplu, cu comenzi de loc complicate, familiare specialiștilor în instruire.

Cuvinte rezervate (Vanilla Pilot pentru Commodore PET). ACCEPT, ACCEPT#, ACCEPT#X, ACCEPT S, APPEND, BEEP, CLEAR HOME, CLOSEFILE, COMPUTE, CREATEFILE, END, EOF, GRAPHICS, HALT, JUMP, KILLFILE, MATCH, OPENFILE, PAUSE, PROBLEM, READ, REMARK, REMARK WRITE, REWIND, TYPE, TYPE HANG, TYPE PRINTER, USE, WAIT, WRITE.

PL/I

Vechi de peste 20 de ani, limbajul PL/I a fost inițial sugerat de către Ad Hoc Committee of SHARE (IBM) în anul 1963. Prima versiune PL/I a apărut în 1966 pentru IBM/360. Prima implementare pe micro subset G a fost realizată în 1980 de Digital Research pentru sistemul de operare CP/M.

Începînd cu această perioadă PL/I a fost implementat pe o mare varietate de microcalculatoare și sisteme de operare inclusiv IBM PC sub sistemul de operare PC-DOS.

Un program PL/I constă din unul sau mai multe blocuri de instrucțiuni, dintre care una poate declara variabile. PL/I este structurat pe blocuri și permite o gestionare eficientă a memoriei. Oferă utilizatorului capacitatea de a declara variabile în blocuri, astfel că spațiul pentru variabile poate fi alocat sau eliberat automat depinzînd de faptul dacă blocul este sau nu activ.

Cuvinte rezervate (PL/I pentru Commodore Super PET și calculatoarele sub CP/M și MS-DOS). ABS, ACOS, ADDR, ALLOCATE, ASCII, ASIN, ATAN, ATAND, AUTO, BASED, BEGIN, BINARY, BIT, BOOL, BUILTIN, CALL, CEIL, CHARACTER, CLOSE, COLLATE, COPY, COS, COSD, COSH, DATE, DECLARE, DECIMAL, DIMENSION, DIVIDE, DO, END, ENTRY, ENVIRONMENT, EXP, EXTERNAL, FILE, FIXED, FLOAT, FLOOR, FORMAT, FREE, GET EDIT, GOTO, HBOUND, IF, % INCLUDE, INDEX, INITIAL, LABEL, LBOUND, LENGTH, LINENO,

LOCK, LOG, LOG 2, LOG 10, MAX, MIN, MOD, NULL, ON, ONCODE, ONFILE, ONKEY, OPEN, PAGENO, POINTER, PROCEDURE, RANK, READ, % REPLACE, RETURN, REVERSE, ROUND, SEARCH, SIGN, SIGNAL, SIN, SIND, SINH, STATIC, STOP, SQRT, SUBSTR, TAN, TAND, TANH, TIME, TRANSLATE, TRIM, TRUNC, UNLOCK, UNSPEC, VARIABLE, VARYING, VERIFY, WRITE.

Prolog

Prolog este un limbaj de nivel foarte înalt, bazat pe ideea utilizării a însăși logicii ca limbaj de programare. Prima implementare a limbajului a avut loc în anul 1972 și a fost realizată de Alain Colmerauer și Philippe Roussel în Marsilia (Franța).

Frumusețea și eleganța limbajului au devenit evidente pentru mulți specialiști în calculatoare (europeni), care apoi au realizat un mare număr de implementări îmbunătățite. După ce scoțianul David Warren a elaborat un compilator comparabil ca eficiență cu LISP-ul oamenii au început să considere serios Prolog-ul ca un limbaj util în Inteligența Artificială fiind un competitor direct al LISP-ului. De fapt există în prezent experimente pe microcalculatoare, atît LISP cît și Prolog care combină cele două limbaje. În anul 1981 a fost lansat în Japonia proiectul generației a V-a de calculatoare, Prolog-ul fiind selectat ca unul din limbajele de bază ale proiectului. Pentru calculatoarele personale a fost realizat Micro Prolog, pe microprocesorul Z80 sub sistemul de operare CP/M.

Cuvinte rezervate (Micro Prolog). add, delete, does, kill, load, list, one, save, sum, term, TIMES.

RPG

Limbajul RPG (Report Programs Generator) a fost creat de IBM în anul 1961. RPG-ul are trei versiuni. RPG II este un limbaj popular, de înalt nivel, flexibil, puternic, compact, convenabil pentru scrierea de programe de gestiune.

În 1983 IBM a anunțat noul sistem/36 mini cu un limbaj prietenos de control RPG II. Dar RPG II este de asemenea disponibil pe calculatoare mari ca IBM/370. De asemenea Hewlett-Packard dispune de o versiune RPG (System/34) pentru minicalculatorul HP 3000.

În octombrie 1978, IBM a introdus System/38 (super mini) pe care s-a implementat RPG III.

Cuvinte rezervate (RPG II). ACQ, ADD, ALTSEQ, AN, AND, BSCA, BEGSR, BITOF, BITON, CHAIN, COMP, CONSOLE, CRT, DEBUG, DISK, DIV, ENDSR, ERASE, EXCPT, EXIT, EXSR, FORCE, GOTO, KEY, KEYBORD, LOKUP, MHHZO, MHLZO, MLLZO, MOVE, MOVEA, MULT, MVR, NEXT, OR, PAGE, PAGE n, PRINTER, READ, REL, RELABL, SET, SETLL, SETOF, SETON, SHTDN, SPECIAL, SQRT, SUB, SUBR xx, TAG, TESTB, TESTZ, TIME, UDATE, UDAY, UMONTH, UYEAR, WORKSTN, XFOOT, Z-ADD, Z-SUB.

SAM 76

SAM 76 este un procesor de liste și șiruri și este adecvat pentru o mare varietate de aplicații interactive și orientate spre utilizator inclusiv programarea în Inteligența Artificială. Limbajul permite o mare portabilitate și are caracteristici comune cu LISP și Forth. SAM 76 a fost creat în Bell Laboratories în anul 1971 de Claude Kagan.

Cuvinte rezervate (pe calculatoarele Apple sub DOS 3.3 și CP/M). ARRAY, branch functions, CONTROL FUNCTION, CONVERSION, FUNCTIONS, GRAPHICS, I/0 functions, logical functions, PARTITION FUNCTION, Storage Functions, Text Division, Text Functions.

Smalltalk

Smalltalk este un limbaj de programare de nivel înalt bazat pe imaginea obiectelor care transmit mesaje unul altuia. A fost elaborat la Xerox's Palo Alto Research Center (PARC) de-a lungul unei perioade de 10 ani, trecînd prin mai multe metamorfoze. Limbajul a fost implementat pe DEC VAX și pe calculatoarele Tektronix 68000, de asemenea pe Apple Lisa. Smalltalk conține 60 clase și sute de mesaje.

Cuvinte rezervate. BLOCK CONTEXT, boolean (subclasses True; False), CHARACTER, Class, Class Description, Collection, DICTIONARY, Integer, Number, object, Sequenceable Collection, STRING, String Functions.

Produse program generalizabile

dBASE II, III, III Plus

dBASE II a fost lansat în 1981 de către Ashton-Tate. El este un sistem de gestiune a bazelor de date relaţionale de dimensiuni mici/medii şi reprezintă un etalon pentru sistemele de gestiune baze de date implementate pe microcalculatoare, calculatoare personale (sub sistemele de operare CP/M şi MS/PC-DOS). O impresionantă listă de funcţii poate manipula aproape orice operaţie asupra unui fişier de date, pornind de la sortarea după chei multiple pînă la crearea şi tipărirea rapoartelor adaptate modificărilor de structură a fişierelor. Gestiunea bazei de date este realizată cu ajutorul unui limbaj de descriere şi manipulare a entităţilor bazei de date şi al unui interpretor (procesor).

Limbajul de programare a comenzilor odată stabilit permite utilizatorului să creeze în cadrul fişierelor de program sisteme dirijate de meniu pentru a facilita manipularea bazei de date, generarea de rapoarte etc.

Structura bazei de date poate fi creată (comanda **CREATE**) în mod direct sau indirect. Conţinutul bazei de date poate fi ordonat fizic sau logic, respectiv prin sortare (crescător/descrescător după un singur cîmp) şi indexare. Activarea unei baze de date se face cu comanda **USE** care deschide fişierul, citeşte dicţionarul datelor (structura) şi prima înregistrare. Încărcarea bazei de date se realizează fie prin inserarea unei înregistrări după înregistrarea curentă sau înaintea ei, fie prin adăugarea la sfîrşitul bazei de date a unei înregistrări (ale cărei cîmpuri au valori neutre) sau a unui grup de înregistrări (introduse în mod interactiv sau dintr-un fişier).

Baza de date poate fi interogată atît global (calculul, afişarea şi memorarea unor cîmpuri, crearea unei baze de date sintetice) cît şi pentru afişarea următoarelor informaţii: structură, conţinut înregistrare/înregistrări, conţinut cîmpuri ce aparţin unei înregistrări etc.

Înregistrările bazei de date pot fi modificate sau şterse. Actualizarea bazei de date, inclusiv adăugările de noi înregistrări întreţine automat şi primul fişier index asociat acesteia, celelalte fişiere index putînd fi de asemenea actualizate.

dBASE II permite activarea unor proceduri, numărul acestora fiind limitat doar de numărul maxim de fişiere ce pot fi deschise simultan de către sistemul de operare utilizat.

Procedurile pot conţine orice comandă dBASE. Limbajul permite implementarea structurilor fundamentale de program: secvenţă, selecţie **(IF-ELSE-ENDIF)**, selecţie generalizată **(DO CASE-ENDCASE)**, iteraţii condiţionate anterior **(DO WHILE-ENDDO)**.

Teoretic, nivelurile de structurare nu sînt limitate.

Prin intermediul fişierelor de text, dBASE permite transmiterea şi preluarea unor informaţii din fişierele BASIC, PASCAL, FORTRAN, PL/1 etc., precizînd în comenzile **APPEND FROM** (preluare) şi **COPY TO** (transmitere) clauza **SDF [DELIMITED]**.

Legătura BASIC-dBASE II (exemplu)

1. Se creează în BASIC (exemplu, BASIC-80) fişierul de structură dată (de exemplu, un fişier de livrări).

```
10 OPEN "O" #1 "LIVRARI.DAT"
20 INPUT COD,CANT,PRET
30 WHILE COD<>99
40    WRITE #1,COD,CANT,PRET
50    INPUT COD,CANT,PRET
60 WEND
70 CLOSE #1
```

2. Se creează aceeaşi structură a fişierului în dBASE II.

```
. CREATE FIDBA
001  COD,N,8
002  CANT,N,4
003  PRET,N,3
004
```

3. Se transferă datele, via BASIC-dBASE II.
. **USE** FIDBA
. **APPEND FROM** LIVRARI.DAT **SDF DELIMITED**
4. Se exploatează în dBASE fişierul transferat (FIDBA).

Legătura dBASE II-BASIC (exemplu)

1. Se creează în dBASE II fişierul de date (exemplu, FIDBA.DBF).
2. Se creează fişierul BASIC (exemplu, LIVRARI.DAT) prin copiere, via dBASE II-BASIC.
. **USE** FIDBA
. **COPY TO** LIVRARI.DAT **SDF DELIMITED**
3. Se exploatează în BASIC fişierul transferat.

Principalele limitări ale lui dBASE II sînt: maximum 32 cîmpuri, maximum 1 000 caractere într-o înregistrare, maximum 65 535 înregistrări, 64 variabile (memorie), 16 fişiere deschise la un moment dat, 99 de caractere într-o expresie de indexare, 24 coloane într-un raport generat cu **REPORT**, 5 expresii într-o comandă **SUM**, 254 de caractere într-o linie de comandă, precizia cîmpurilor numerice 10 cifre, cel mai mare număr $1{,}8 \times 10^{63}$.

Funcţiile dBASE pot fi extinse cu ajutorul unor rutine scrise în limbaj de asamblare.

Prima versiune a lui dBASE a fost scrisă în ASSEMBLER 8080, în 1978 de Wayne Ratliff, proiectant de sistem la Jet Propulsion Laboratory (California).

După 1983 limbajul a fost scris pentru MS-DOS şi CP/M pe calculatoare de 16 biţi. În anul 1984 a apărut dBASE III.

Principalele limitări ale lui dBASE III sînt: lucrează sub MS-DOS pe 16 biţi, maximum 128 de cîmpuri, maximum un miliard de înregistrări per fişier, maximum 4 000 octeţi per înregistrare, maximum cinci tipuri de cîmpuri C, M, N, L, D, 10 fişiere de date, 7 fişiere de index, 15 fişiere de orice tip deschise simultan.

În anul 1985 a fost lansat de către Ashton-Tate, dBASE III Plus ale cărui limite sînt: maximum un miliard înregistrări într-o bază de date; record size – 4 000 bytes în .dbf file şi 512 kilobytes în .dbt file; maximum 128 cîmpuri într-o înregistrare; maximum cinci tipuri de cîmpuri – C (character), D (Date), L (Logical), M (Memo), N (Numeric); maximum 254 caractere într-un cîmp de tip C, maximum 8 bytes într-un cîmp de tip D, 1 byte într-un cîmp de tip L, maximum 5 000 bytes într-un cîmp de tip M, maximum 19 bytes într-un cîmp de tip N; maximum 256 de variable; maximum 254 caractere pe linia de comandă. În prezent, este operaţional dBASE IV.

Lista comenzilor dBASE II, III, III Plus[1]

?/??
@
@ . . . TO (*)
ACCEPT
APPEND
APPEND FROM
ASSIST (***)
AVERAGE (***)
BROWSE
CALL (*)
CANCEL
CHANGE
CLEAR
CLEAR ALL (***)
CLEAR FIELDS (*)
CLEAR GETS
CLEAR MEMORY (***)
CLEAR TYPEAHEAD (*)
CLOSE (***)
CONTINUE
COPY
COPY FILE (***)
COPY STRUCTURE
COPY STRUCTURE EXTENDED (*)
COUNT
CREATE
CREATE LABEL (***)
CREATE QUERY (*)
CREATE REPORT (***)
CREATE SCREEN (*)
CREATE VIEW (*)
CREATE VIEW FROM ENVIRONMENT (*)
DELETE
DELETE FILE (****)
DIR (***)
DISPLAY
DISPLAY HISTORY (*)
DISPLAY MEMORY
DISPLAY STATUS

---

[1] Comenzile nemarcate aparţin celor trei limbaje; * (dBASE III Plus); ** (dBASE II); *** (dBASE III+dBASE III Plus); **** (dBASE II+dBASE III Plus).

DISPLAY STRUCTURE
DISPLAY FILES (**)
DO
DO CASE
DO WHILE

EDIT
EJECT
ELSE
ENDCASE
ENDDO
ENDIF
ENDTEXT
ERASE
EXIT (***)
EXPORT (*)

FIND

GO
GOTO

HELP

IF
IMPORT (*)
INDEX
INPUT
INSERT

JOIN

LABEL (***)
LIST HISTORY (*)
LIST MEMORY
LIST STATUS
LIST STRUCTURE
LIST FILES (**)
LOAD (*)
LOCATE
LOOP

MODIFY COMMAND
MODIFY STRUCTURE
MODIFY LABEL (***)
MODIFY QUERY (*)
MODIFY REPORT (***)
MODIFY SCREEN (*)
MODIFY VIEW (*)

NOTE/*

ON ERROR/ESCAPE/KEY (*)

PACK
PARAMETERS (***)
PRIVATE (*)
PROCEDURE (*)
PUBLIC (*)
PEEK (**)
POKE (**)

QUIT

READ

RECALL
REINDEX (*)
RELEASE
REMARK (**)
RENAME
REPLACE
REPORT
RESET (**)
RESTORE
RESUME (*)
RETRY (*)
RETURN
RUN / ! (*)

SAVE
SEEK (***)
SELECT
SET
SET ALTERNATE
SET BELL
SET CARRY
SET CATALOG (*)
SET CATALOG TO (*)
SET CENTURY (*)
SET COLON (**)
SET COLOR
SET CONFIRM
SET CONSOLE
SET DATE (** **)
SET DEBUG
SET DECIMALS (***)
SET DEFAULT
SET DELETED
SET DELIMITER (***)
SET DEVICE (***)
SET DOHISTORY (*)
SET ECHO
SET EJECT (**)
SET ESCAPE
SET EXACT
SET FIELDS (*)
SET FIELDS TO (*)
SET FILTER (***)
SET FIXED (***)
SET FORMAT
SET FUNCTION (***)
SET HEADING
SET HELP (***)
SET HISTORY (*)
SET HISTORY TO (*)
SET INDEX
SET INTENSITY
SET LINKAGE (**)
SET MARGIN
SET MEMOWIDTH TO (*)
SET MENU (***)
SET MESSAGE TO (*)
SET ORDER (*)
SET PATH (***)
SET PRINT
SET PRINTER (*)
SET PROCEDURE (***)
SET RAW (**)
SET RELATION (***)

SET SAFETY (***)
SET SCOREBOARD (*)
SET SCREEN (**)
SET STATUS (*)
SET STEP
SET TALK
SET TITLE (*)
SET TYPEAHEAD TO (*)
SET UNIQUE (***)
SET VIEW TO (*)
SKIP
SORT
STORE

SUM
SUSPEND (*)

TEXT
TOTAL
TYPE (*)

UPDATE
USE

WAIT

ZAP (*)

Lista funcţiilor dBASE II, III, III Plus[1]

$ (**)
! (**)
&
@ (**)
* (**)
(**)

ABS (*)
ASC (***)
AT (***)

BOF (***)

CDOW (***)
CHR
CMONTH (***)
COL (***)
CTOD (***)

DATE
DAY (***)
DBF (*)
DELETED (***)
DISKSPACE (*)
DOW (***)
DTOC (***)

EOF
ERROR (*)
EXP (***)

FIELD (*)
FILE
FKLABEL (*)
FKMAX (*)
FOUND (*)

GETENV (*)

IIF (*)
INKEY (*)
INT
ISALPHA (*)
ISCOLOR (*)
ISLOWER (*)
ISUPPER (*)

LEFT (*)
LEN
LOG (***)
LOWER (***)
LTRIM (*)
LUPDATE (*)

MAX (*)
MESSAGE (*)
MIN (*)
MOD (*)
MONTH (***)

NDX (*)

OS (*)

PCOL (***)
PROW (***)

RANK (****)
READKEY (*)
RECCOUNT (*)
RECNO (***)
RECSIZE (*)
REPLICATE (*)
RIGHT (*)
ROUND (***)
ROW (***)
RTRIM (*)

SPACE (***)
SQRT (***)
STR
STUFF (*)
SUBSTR (***)

TIME (***)
TRANSFORM (*)
TRIM
TYPE

UPPER (***)

VAL
VERSION (*)
YEAR (***)

---

[1] Funcţiile nemarcate aparţin celor trei limbaje; * (dBASE III Plus); ** (dBASE II); *** (dBASE III+dBASE III Plus); **** (dBASE II+dBASE III Plus).

Visi Calc

Visi Calc (VISUAL CALCULATOR) este primul program disponibil pe Apple II fiind utilizat în analiza financiară.

Cuvinte rezervate (Apple, Commodore PET). @ABS(n), @INT(n), @SQRT(n), @EXP(n), @LOG10(n), @LN(n), @SIN(n), @COS(n), @TAN(n), @ASIN(n), @ACOS(n), @ATAN(n), @NPV, @NA, @ERROR, @PI, @LOOKUP, @SUM(LIST), @MAX(LIST), @MIN(LIST), @COUNT(LIST), @AVERAGE(LIST), /B, /C, /D, /F, /G, GOTO>, I, LABEL ENTRY", /M, /P, /R, /S, /T, /V, /W.

Super Calc

Super Calc este înrudit cu Visi Calc și se folosește, de asemenea în analizele financiare.

Cuvinte rezervate (sub CP/M). ABS, AND, AVERAGE, /C, COUNT, /D, ERROR, EXP, /G, /I, INT, IF, LOOKUP, /L, /M, NOT, /O, OR(N1, N2) /Q, /P, /R, /S, /T, /U, /Z, /W.

Multi Plan

Multi Plan este un limbaj de generația a II-a pentru rapoarte pe 65 coloane și 255 de rînduri. A fost creat de Micro Soft. Multi Plan este operațional aproape pe orice calculator de 8, 16, 32 biți sub CP/M, UNIX, MS-DOS. Limbajul oferă numeroase posibilități ca rapoartele pe care le generează (structura de comenzi a meniului este tot timpul afișată) să se poată adapta pentru numeroase configurații hard.

Cuvinte rezervate. ALPHA, BLANK, COPY, DELETE, EDIT, FORMAT, GOTO, HELP, INSERT, LOCK, MOVE, NAME, OPTIONS, PRINT, QUIT, SORT, TRANSFER, VALUE, WINDOW, eXTERNAL.

Lotus 1–2–3

Lotus 1–2–3 a fost introdus în anul 1983 de către corporația Lotus din Boston, pentru generarea de rapoarte. 1–2–3 indică cele trei componente ale limbajului. A fost scris special pentru IBM PC și alte calculatoare sub MS-DOS. Poate accesa fișiere proprii dar și din dBASE II sau Visi Calc.

Cuvinte rezervate. ABS, CALC, COPYand MOVE, DATA, EDIT, GOTO, GRAPH, HELP, QUERY, NAME, RANGE, RANGE ERASE, WINDOW, WORKSHEET.

Symphony

Este o extensie a lui Lotus 1–2–3. A fost lansat în anul 1984 și este disponibil pe IBM PC/XT/AT. Symphony cere minimum 320 k în RAM fiind deosebit de complex (cu bază de date, procesor de cuvinte, macroprograme etc.).

Framework

Framework a fost lansat în anul 1984 de către Aston-Tate. Se bazează pe multe componente de program realizînd: procesare de cuvinte, gestionare baze de date, calcule pe formulare afișate electronic, grafică comercială, sublinieri etc. Cadrele tridimensionale ce apar pe ecran pot fi: deschise, imbricate, expandate/contractate și localizate oriunde pe ecran.

FRED

Formulele utilizate de către componenta de formulare afișate electronic de Framework sînt toate parte componentă a limbajului de programare FRED ce este inclus în pachetul Framework.

Cuvinte rezervate. **@ABS, @ACOS, @VG, @ASIN, @ATAN, @ATAN2, @AND, @BM, @BEEP, @BUSINESS, @CEILING, @COS, @CHOOSE, @COUNT, @CHR, @CURRENCY, @DATE, @DATE1, @DATE2, @DATE3, @DATE4, @DATETIME, @DIFFDATE, @DBASEFILTER, @DRAW, @DRAWGRAPH, @DECIMAL, @DOLLAR, @DISPLAY, @EXECUTE, @ECHO, @EXP, @ERASEPROMPT, @FV, @FLOOR, @FC, @FL, @FP, @FR, @FILL, @GETENV, @GETFORMULA, @GET, @HC, @HF, @HL, @HP, @HR, @HLOOKUP, @HIDE, @IRR, @ITEM1 ... @ITEM16, @ITEM, @INTEGER, @ITEMCOUNT, @IF, @ISNA, @ISERR, @ISABEND, @ISALPHA, @ISNUMERIC, @INT, @INPUTLINE, @KEY, @KEYFILTER, @KEYNAME, @KP, @LIST, @LOCAL, @LOG, @LL, @LEN, @MEMAVAIL, @MIRR, @MOD, @MAX, @MIN, @MID, @MILLI, @MENU, @NPV, @NOT, @NEXTKEY, @NP, @NEXT, @NATIONALIZE, @OR, @PRINTRETURN, @PMT, @PV, @PERFORMKEYS, @PI, @PL, @PO, @PR, @PRINT, @PN, @PUT, @POUND, @PROMPT, @RESULT, @RUN, @RETURN, @RAND, @ROUND, @RESET, @REPT, @SELECT, @SET, @SETDIRECTORY, @SETDRIVE, @SETFORMULA, @SETMACRO, @SETSELECTION, @SIGN, @SIN, @SQRT, @ST, @SK, @SP, @STD, @SUM, @SCIENTIFIC, @TRACE, @TIME, @TIME1, @TIME2, @TIME3, @TODAY, @TAN, @TM, @THOUSANDS, @UNIT, @UNHIDE, @VLOOKUP, @VAR, @VALUE, @YEN, @WHILE, @WRITETEXTFILE.**

□ Vocabular comparativ al limbajelor de programare de nivel înalt: BASIC*, Ada, Algol, APL, C, COBOL, COMAL, Forth, PC Forth, Fortran, LISP, Logo, Modula-2, Pascal, Pilot, PL/I, Prolog, RPG, SAM 76, Smalltalk și al produselor program dBASE II, III, III Plus, Visi Calc, Super Calc, Multi Plan, Lotus 1–2–3, Symphony, Framework

A

ABORT
Forth

ABORT "
PC Forth

ABS
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
CBASIC
ZBASIC
COMAL
Forth
Lotus 1–2–3
Modula 2
Pascal
PL/I
Super Calc
dBASE III Plus

@ABS
FRED

@ABS (n)
Visi Calc

ACCEPT
BASIC-TI (extins)
COBOL
Pilot

ACCEPT#
Pilot

ACCEPT S
Pilot

ACCEPT
dBASE II, III, III Plus

ACCEPT# x
Pilot

ACOS
PL/I
Super Calc

@ACOS
FRED

---

* BASIC-aMIC, BASIC-PRAE, BASIC HC-85, TIM S, SPECTRUM, BASIC-AMSTRAD, BASIC-COMMODORE, BASIC-80, BASIC-PLUS, ABASIC, GW-BASIC, MBASIC 86, ZBASIC, CBASIC, S-BASIC, BASIC-APPLESOFT, BASIC-ATARI; BASIC-TI (extins).

ACOS (n)
Visi Calc
ACQ
RPG
ACS
BASIC HC-85, TIMS, SPECTRUM
BASIC PLUS
add
Prolog
ADD
COBOL
RPG
ADDR
PC Forth
PL/I
ADDR (Var $)
BASIC-ATARI
AFTER
BASIC-AMSTRAD
AGAIN
PC Forth
ALL
COBOL
ALLOAD
BASIC-PRAE
ALLOCATE
PL/I
ALLOC
C
ALLOF
Logo
ALLOT
Forth
PC Forth
ALPHA
Multi Plan
ALSO
PC Forth
AMERGE
BASIC-PRAE
AN
RPG
AND
BASIC-PRAE
BASIC HC-85, TIMS, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
ZBASIC
COBOL
COMAL
Forth
Modula-2
Pascal
PC Forth
RPG
Super Calc
@AND
FRED
AND&&
C

ANYOF
Logo
APPEND
BASIC-PLUS
LISP
dBASE II, III, III Plus
APPEND FROM
dBASE II, III, III Plus
APPLY
LISP
ARCTAN
Modula-2
Pascal
ARRAY
LISP
Modula-2
Pascal
ARRAY
LISP
Modula-2
Pascal
SAM 76
AS
BASIC-PLUS
ASAVE
BASIC-PRAE
ASC
dBASE III, III Plus
BASIC-PRAE
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
ABASIC
GW-BASIC
CBASIC
ZBASIC
ASCII
BASIC-PLUS
PC Forth
PL/I
ASIN
PL/I
Super Calc
@ASIN
FRED
@ASIN (n)
Visi Calc
ASN
BASIC HC-85, TIM S, SPECTRUM
BASIC-PLUS
ASSEMBLER
PC Forth
ASSIGN
COBOL
Fortran
ASSIST
dBASE III, III Plus
AT
dBASE III, III Plus
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM

AT END
COBOL
ATAN
PL/I
Super Calc
@ATAN
FRED
@ATAN 2
FRED
ATAND
PL/I
@ATAN (n)
Visi Calc
ATN
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
CBASIC
ZBASIC
BASIC-APPLESOFT
COMAL
ATOF
C
ATOI
C
ATOM
LISP
ATTR
BASIC HC-85, TIM S, SPECTRUM
ATTRIBUTE
PC Forth
Author
COBOL
AUTO
BASIC-PRAE
BASIC-AMSTRAD
BASIC-80
ABASIC
GW-BASIC
PL/I
AVERAGE
dBASE III, III Plus
Super Calc
AVERAGE (LIST)
Visi Calc
@AVG
FRED

B

/B
Super Calc
Visi Calc
BACK (BK)
Logo
BACKGROUND
PC Forth
BACKGROUND (BG)
Logo
BACKSPACE
Fortran
BASE
BASIC-80
ABASIC
Forth
PC Forth
BASED
PL/I
B/BUF
PC Forth
BEL
ABASIC
BEEP
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
GW-BASIC
Pilot
ZBASIC
@beep
FRED
BEGIN
S-BASIC
Forth
Modula-2
Pascal
PC Forth
PL/I
BEGSR
RPG
BELL
PC Forth
BIN
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BINARY
PC Forth
PL/I
C
BIT
Algol
C
PL/I
BITOF
RPG
BITON
RPG
BITSET
Modula-2
bitwise AND&
C
bitwise inclusive OR
C
BL
PC Forth
BLANK
COBOL
Multiplan
PC Forth
BLK
Forth
PC Forth

CALL SAY
BASIC-TI (extins)
CALL SPRITE
BASIC-TI (extins)
CALL VERSION
BASIC-TI (extins)
CANCEL
dBASE II, III, III Plus
CAP (ch)
Modula-2
CAPS
PC Forth
CAR
LISP
CARDINAL
Modula-2
CASE
S-BASIC
Algol
C
COMAL
Modula-2
Pascal
CASE:
PC Forth
CAT
BASIC-AMSTRAD
CATALOG
Logo
CATCH
LISP
CDBL
BASIC-80
ZBASIC
CDOW
dBASE III, III Plus
CDR
LISP
CEIL
PL/I
@CEILING
FRED
cfree
C
CHAIN
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
COMAL
RPG
CHANGE
dBASE II, III, III Plus
char
C
CHAR
S-BASIC
Algol
Modula-2
Pascal
CHARACTER
Fortran
Smalltalk
PL/I
@CHOOSE
FRED
CHDIR
GW-BASIC
CHR
dBASE II, III, III Plus
ABASIC
COMAL
Modula-2
Pascal
CHRA
ABASIC
@CHR
FRED
CHR$
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
GW-BASIC
CBASIC
ZBASIC
CHR$(num)
BASIC-APPLESOFT
CINT
BASIC-AMSTRAD
BASIC-80
ABASIC
GW-BASIC
ZBASIC
CIRCLE
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
GW-BASIC
ZBASIC
CLEAR
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-80
ABASIC
GW-BASIC
BASIC-APPLESOFT
ZBASIC
dBASE II, III, III Plus
CLEAR ALL
dBASE III, III Plus
CLEAR GETS
dBASE II, III, III Plus
CLEAR FIELDS
dBASE III Plus
CLEAR HOME
Pilot
CLEAR MEMORY
dBASE III, III Plus
CLEARSCREEN
PC Forth
CLEARSCREEN(cs)
Logo

CONST
Modula-2
Pascal
CONSTANT
Ada
Forth
PC Forth
$CONSTANT
S-BASIC
2 CONSTANT
PC Forth
CONSTANT%
CBASIC
CONSTRAINT
Ada
CONT
BASIC-PRAE
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
ABASIC
GW-BASIC
BASIC-APPLESOFT
ZBASIC
CONTEXT
PC Forth
CONTEXT CLAUSE
Ada
CONTINUE
BASIC HC-85, TIM S, SPECTRUM
BASIC-PLUS
dBASE II, III, III Plus
Fortran
CONTROL
PC Forth
CONTROL FUNCTION
SAM 76
CONVERSION
SAM 76
CONVERT
Forth
PC Forth
COPY
BASIC HC-85, TIM S, SPECTRUM
Multiplan
PL/I
COPY CHR$
BASIC-AMSTRAD
COPY and MOVE
Lotus 1–2–3
COPY function
C
COPY FILE
dBASE III, III Plus
COPY
dBASE II, III, III Plus
COPY STRUCTURE EXTENDED
dBASE III Plus
COPY STRUCTURE
dBASE II, III, III Plus
CORRESPONDING (CORR)
COBOL
COS
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
ZBASIC
CBASIC
BASIC-APPLESOFT
COMAL
Modula-2
Pascal
Super Calc
PL/I
2@COS
FRED
COSD
PL/I
COSH
PL/I
2@COS(n)
Visi Calc
COUNT
BASIC-PLUS
dBASE II, III, III Plus
Forth
PC Forth
Super Calc
2@COUNT
FRED
2@COUNT (LIST)
Visi Calc
CPOS
ABASIC
CR
Forth
2 create
C
CREAL
BASIC-AMSTRAD
CREATE
S-BASIC
CBASIC
dBASE II, III, III Plus
Forth
CREATE LABEL
dBASE III, III Plus
CREATEFILE
Pilot
CREATE QUERY
dBASE III Plus
CREATE REPORT
dBASE III, III Plus
CREATE SCREEN
dBASE III Plus
CREATE VIEW
dBASE III Plus
CREATE VIEW FROM ENVIRONMENT
dBASE III Plus
CRESET
PC Forth

D

DEC$
BASIC-AMSTRAD
DEC (x, n)
Modula-2
decrement operator
C
DEF
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
CBASIC
DEFDBL
BASIC-80
ABASIC
GW-BASIC
ZBASIC
DEF FN
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
ABASIC
BASIC-APPLESOFT
ZBASIC
DEFINITION
Modula-2
DEFINITIONS
Modula-2
DEFINT
BASIC-AMSTRAD
BASIC-80
ABASIC
GW-BASIC
ZBASIC
DEFMACRO
LISP
DEFREAL
BASIC-AMSTRAD
DEFSEG
ZBASIC
DEFSIGN
GW-BASIC
DEFSNG
BASIC-80
DEFSTR
BASIC-AMSTRAD
BASIC-80
ABASIC
GW-BASIC
ZBASIC
LISP
DEFUN
LISP
DEF USR
BASIC-80
ZBASIC
DEG
BASIC-AMSTRAD
BASIC-ATARI
DELETED
dBASE III, III Plus
delete
Prolog
DELETE
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
dBASE II, III, III Plus
Multi Plan
DELETE FILE
dBASE II, III Plus
DELIMITED
COBOL
DENOTE
Ada
DEPENDING
COBOL
DEPTH
Forth
PC Forth
DERIVED TYPE
Ada
DESIGNATE
Ada
DERR
BASIC-AMSTRAD
DICTIONARY
Smalltalk
DIF$
BASIC-PLUS
@DIFFDATE
FRED
DIGIT
ABASIC
PC Forth
DI
BASIC-AMSTRAD
DIM
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
CBASIC
ZBASIC
COMAL
DIMENSION
ABASIC
Fortran
PL/I
DIR
PC Forth
dBASE III, III Plus
DIRECTORY NODE
Ada
DISCRETE TYPE
Ada

E

@ECHO
FRED
EDIT
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-80
ABASIC
GW-BASIC
ZBASIC
Logo
Lotus 1–2–3
Multi Plan
edit
dBASE II, III, III Plus
EDITOR
PC Forth
EI
BASIC-AMSTRAD
EJECT
dBASE II, III, III Plus
ELIF
COMAL
ELSE
BASIC-PRAE
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
dBASE II, III, III plus
ABASIC
GW-BASIC
Algol
COBOL
COMAL
Forth
Modula-2
Pascal
PC Forth
ELSE IF
Modula-2
EMIT
Forth
PC Forth
EMPTY-BUFFERS
PC Forth
END
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
ZBASIC
CBASIC
Algol
Fortran
Logo
Modula-2
Pascal
Pilot
PL/I
END?
PC Forth
ENDCASE
COMAL
END-CODE
PC Forth
ENDFILE
Fortran
ENDIF
ABASIC
COMAL
ENDPROC
ABASIC
COMAL
ENDSR
RPG
ENDWHILE
ABASIC
Fortran
ENDTEXT
dBASE III, III Plus
ENT
BASIC-AMSTRAD
ENTER
BASIC HC-85, TIM S, SPECTRUM
BASIC-ATARI
ENTITY
Ada
ENTRY
Ada
Fortran
PC Forth
PL/I
ENUMERATING TYPE
Ada
ENV
BASIC-AMSTRAD
ENVIRONMENT
PL/I
ENVIRONMENT DIVISION
COBOL
ENVIRON
GW-BASIC
EOF
dBASE II, III, III Plus
BASIC-80
BASIC-AMSTRAD
ABASIC
GW-BASIC
ZBASIC
Pilot
EOL
Modula-2
EQL
LISP
EQV
ABASIC
GW-BASIC, BASIC-80
S-BASIC, BASIC-PLUS
EQUAL
LISP
EQUIVALENCE
Fortran

PC Forth
@FC
FRED
fclose
C
FD
COBOL
FENCE
PC Forth
FEND
CBASIC
FFIX
S-BASIC
FIELD
dBASE III Plus
BASIC-80
BASIC-PLUS
GW-BASIC
ZBASIC
FILE
dBASE II, III, III Plus
BASIC-PLUS
CBASIC
Pascal
PL/I
Lotus 1–2–3
FILE CONTROL
COBOL
FILE NODE
ADA
FILE SIZE
BASIC-PLUS
FILES
GW-BASIC
ZBASIC
FILL
BASIC-AMSTRAD
Forth
PC Forth
@FILL
FRED
FILLER
COBOL
FIND
Forth
PC Forth
dBASE II, III, III Plus
FINT
S-BASIC
FIRST
Logo
PC Forth
FIX
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
ZBASIC
FKLABEL
dBASE III Plus
FKMAX
dBASE III Plus
@FL
FRED
FLASH
BASIC HC-85, TIM S, SPECTRUM
BASIC-APPLESOFT
float
C
FLOAT
CBASIC
Modula-2
PL/I
FLOOR
PL/I
@FLOOR
FRED
FLUSH
Forth
PC Forth
FN
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
FNEND
BASIC-PRAE
BASIC-PLUS
fopen
C
FOR
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
CBASIC
ZBASIC
Algol
COMAL
Modula-2
Pascal
FOR . . . NEXT
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
CBASIC
ZBASIC
FOR . . . NEXT NESTING
BASIC-TI (extins)

BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
Algol
COBOL
COMAL
Fortran
Lotus 1–2–3
Multi Plan
Pascal
PL/I
RPG
dBASE II, III, III Plus

GOTO
Visi Calc

GSINPUT
BASIC-aMIC

GRAPHICS PAPER
BASIC-AMSTRAD

GRAPHICS PEN
BASIC-AMSTRAD

GRAPH
Lotus 1–2–3

GRAPHICS
Pilot
SAM

GREATER
COBOL

GWBASIC
GW-BASIC

H

HALT
Modula-2
Pilot

HBOUND
PL/I

@HC
FRED

HCOLOR
BASIC-APPLESOFT

HEADING
COBOL
Logo

HEAP
Algol

HELP
BASIC-PLUS
Lotus 1–2–3
Multi Plan
dBASE II, III, III Plus

HERE
Forth
PC Forth

HEX
ABASIC
GW-BASIC
PC-Forth

HEX$
BASIC-AMSTRAD

@HF
FRED

HGR
BASIC-APPLESOFT

HGR2
BASIC-APPLESOFT

HIDE
PC Forth

@HIDE
FRED

HIDETURTLE (HT)
Logo

HIGH
Modula-2

HIGH-VALUES
COBOL

HIMEM
BASIC-AMSTRAD
BASIC-APPLESOFT

@HL
FRED

HLD
PC Forth

@HLOOKUP
FRED

HOLD
Forth
PC Forth

HOME
BASIC-APPLESOFT
Logo

@HP
FRED

HPLOT
BASIC-APPLESOFT

@HR
FRED

HTAB
BASIC-APPLESOFT

I

/I
Visi Calc

I
Forth
PC Forth
Visi Calc

IDN
BASIC-aMIC

IDENTIFICATION DIVISION
COBOL

IF
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80

INSTALLATION
COBOL
INSTR
BASIC-PRAE
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
ZBASIC
Int
C
INT
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-ATARI
CBASIC
BASIC-APPLESOFT
ZBASIC
Algol
COMAL
Super Calc
dBASE II, III, III Plus
@INT
FRED
INTEGER
Fortran
Modula-2
Pascal
Integer
Smalltalk
@INTEGER
FRED
INTERPRET
PC Forth
@INT (n)
Visi Calc
INTO
COBOL
INV
BASIC-aMIC
INVERSE
BASIC HC-85, TIM S, SPECTRUM
BASIC-APPLESOFT
IOCTL
GW-BASIC
IOCTL$
GW-BASIC
@IRR
FRED
@ISABEND
FRED
ISALPHA
dBASE III Plus
ISCOLOR
dBASE III Plus
@ISALPHA
FRED
@ISERR
FRED
ISLOWER
dBASE III Plus
ISUPPER
dBASE III Plus
@ISNA
FRED
@ISNUMERIC
FRED
@ITEM
FRED
@ITEMCOUNT
FRED
@ITEM 1 . . . ITEM 16
FRED
Itoa
C

J

J
Forth
PC Forth
JOY
BASIC-AMSTRAD
JOIN
dBASE II, III, III Plus
JUMP
Pilot
JUSTIFIED
COBOL

K

K
PC Forth
KEY
BASIC-AMSTRAD
GW-BASIC
ZBASIC
Forth
PC Forth
RPG II
@KEY
FRED
KEYBORD
RPG II
KEY DEF
BASIC-AMSTRAD
@KEYFILTER
FRED
KEY LIST
GW-BASIC
KEY OFF
GW-BASIC
ZBASIC
KEY ON
GW-BASIC
ZBASIC
KEY STOP
GW-BASIC

ABASIC
GW-BASIC
BASIC-ATARI
dBASE II, III, III Plus
LISP
PC Forth

LIST HISTORY
dBASE III Plus

@LIST
FRED

$LIST
S-BASIC

LIST "P"
BASIC-ATARI

LISTP
LISP

LITERAL
Forth
PC Forth

@LL
FRED

LLIST
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-80
GW-BASIC
ZBASIC

Ln
Super Calc

LN
BASIC HC-85, TIM S, SPECTRUM
ABASIC
Modula-2
Pascal

@LN(n)
Visi Calc

load
Prolog

LOAD
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
Forth
dBASE III Plus
PC Forth

$LOADPT
S-BASIC

LOC
ZBASIC
GW-BASIC
BASIC-80
Algol

LOCATE
BASIC-AMSTRAD
GW-BASIC
BASIC-ATARI
ZBASIC
S-BASIC
dBASE II, III, III Plus

LOCATION
S-BASIC

LOCK
Multi Plan
PL/I

LOF
GW-BASIC
ZBASIC

LOG
dBASE III, III Plus
BASIC-aMIC
BASIC-PRAE
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
BASIC-APPLESOFT
CBASIC
COMAL
PL/I
ZBASIC

@LOG
FRED

LOG 2
PL/I

LOG 10
BASIC-AMSTRAD
BASIC-PLUS
ABASIC
PL/I
Super Calc

LOGICAL
Fortran

@LOG 10 (n)
Visi Calc

LOKUP
RPG

LOMEM
BASIC-APPLESOFT

long
C

LONG
Algol

LOOKUP
Super Calc

@LOOKUP
Visi Calc

LOOP
dBASE II, III, III Plus
Forth
Modula-2
PC Forth

LOWER
dBASE III, III Plus

LOWER$
BASIC-AMSTRAD

LOW-VALUES
COBOL

LNULL
BASIC-PRAE

LPOS
BASIC-80

GW-BASIC
ZBASIC
MKI$
BASIC-80
GW-BASIC
ZBASIC
MKS$
BASIC-80
GW-BASIC
ZBASIC
MLHZO
RPG
MLLZO
RPG
MOD
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
dBASE III Plus
COMAL
Forth
FRED
Pascal
PC Forth
PL/I
MODE
BASIC-AMSTRAD
modo
PC Forth
MODIFY COMMAND
dBASE II, III, III Plus
MODIFY STRUCTURE
dBASE II, III, III Plus
MODIFY LABEL
dBASE III, III Plus
MODIFY QUERY
dBASE III Plus
MODIFY REPORT
dBASE III, III Plus
MODIFY SCREEN
dBASE III Plus
MODIFY VIEW
dBASE III Plus
MODULE
Modula-2
MONITOR FUNCTIONS
SAM 76
MONTH
dBASE III, III Plus
MOUNT
PC Forth
MOVE
BASIC-aMIC
BASIC-AMSTRAD
BASIC-80
COBOL
Multi Plan
PC Forth
RPG
MOVEA
RPG
MOVER
BASIC-AMSTRAD
MULT
RPG
MULTAB
Algol
MULTIPLY
COBOL
MVR
RPG

N

@NA
Visi Calc
NAME
BASIC-80
GW-BASIC
ZBASIC
@NATIONALIZE
FRED
NCONC
LISP
NDX
dBASE III Plus
NEGATE
Forth
NEW
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
Modula-2
Pascal
NEXT
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
BASIC-ATARI
ZBASIC
RPG
@NEXT
FRED
@NEXTKEY
FRED
NEXT SENTENCE
COBOL
NEXT WITHOUT FOR
BASIC-TI (extins)
NIL
Modula-2
Pascal
NODRAW
Logo

OPENFILE
Pilot
OPERATION
Ada
OPERATOR
Ada
OPTION BASE
BASIC-80
ABASIC
GW-BASIC
ZBASIC
OPTIONS
Multi Plan
OR
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-TI (extins)
ZBASIC
COBOL
COMAL
Forth
Modula-2
Pascal
PC Forth
RPG
Super Calc
@OR
FRED
ORD
COMAL
Modula-2
Pascal
ORDER
PC Forth
ORGANIZATION
COBOL
ORIGIN
BASIC-AMSTRAD
OS
dBASE III Plus
OTHERWISE
COMAL
OUT
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-80
GW-BASIC
OUTDEV
Logo
OUTPUT
BASIC-PLUS
COMAL
Logo
Pascal
OVER
BASIC HC-85, TIM S, SPECTRUM
Forth
OVERLOADING
Ada

P

/P
Super Calc
Visi Calc
PACK
dBASE II, III, III Plus
Pascal
PACKAGE
Ada
PACKED
Pascal
PAD
Forth
PC Forth
PADDLE
BASIC-ATARI
LOGO
PADDLEBUTTON
LOGO
PADDLECOLOR
Logo
PAGE
Pascal
RPG
$PAGE
S-BASIC
PAGE-COUNTER
COBOL
PAGE n
RPG
PAGENO
PL/I
PAINT
GW-BASIC
ZBASIC
PAPER
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
PARAMETER
Ada
Fortran
PARAMETERS
dBASE III, III Plus
PARENTNODE
Ada
PARENT TYPE
Ada
PARTITION FUNCTION
SAM 76
PAUSE
BASIC HC-85, TIM S, SPECTRUM
Fortran
Pilot
PCOL
dBASE III, III Plus
PEEK
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM

@PRINT
FRED
PRINT 1
Logo
PRINTER
PC Forth
RPG
PRINTER-INIT
PC Forth
PRINTER-OUT
PC Forth
PRINTER-STATUS
PC Forth
printif
C
PRINTING
PC Forth
PRINTOUT
Logo
PRINTOUT TITLES
Logo
@PRINTRETURN
FRED
PRIVATE
dBASE III Plus
PRIVATE PART
Ada
PRIVATE TYPE
Ada
PROBLEM
Pilot
PROC
Algol
COMAL
PROCEDURE
S-BASIC
Modula-2
Pascal
PL/I
dBASE III Plus
PROCEDURE CALL
Ada
PROCEDURE-DIVISION
COBOL
PROG 1
LISP
PROGNA
LISP
PROGRAM
Ada
PROGRAM LIBRARY
Ada
@PROMPT
FRED
PROPS
LISP
PROW
dBASE III, III Plus
PSET
GW-BASIC
ZBASIC
PTRIG
BASIC-ATARI
PUBLIC
dBASE III Plus
PUT
BASIC-aMIC
BASIC-80
BASIC-PLUS
GW-BASIC
ZBASIC
LISP
@PUT
FRED
putc
C
putchar
C
@PV
FRED

Q

/Q
Super Calc
QUALIFIED
Modula-2
QUALIFIED EXPRESSION
Ada
QUERY
Lotus 1–2–3
PC Forth
QUILT
Forth
QUIT
dBASE II, III, III Plus
Multi Plan
@QUITMENU
FRED
QUOTE
LISP

R

/R
Super Calc
Visi Calc
R>
Forth
R@
Forth
R#
PC Forth
RAD
BASIC-AMSTRAD
BASIC-PLUS
BASIC-ATARI
@RAND
FRED
RANDOM
BASIC-PLUS
Logo
RANDOM NUMBER
SAM 76
RANDOMIZE
BASIC-PRAE

RENAMES
COBOL
RENDEZVOUS
Ada
RENUM
BASIC-AMSTRAD
BASIC-80
ABASIC
GW-BASIC
ZBASIC
RENUMBER
BASIC-PRAE
REPEAT
S-BASIC
COMAL
Forth
Logo
Modula-2
Pascal
PC Forth
REPLACE
dBASE II, III, III Plus
%REPLACE
PL/I
REPLICATE
dBASE III Plus
REPORT
dBASE II, III, III Plus
REPRESENTATION CLAUSE
Ada
@REPT
FRED
REQUEST
Logo
RESET
GW-BASIC
dBASE II
@RESET
FRED
REST
LISP
RESTORE
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
CBASIC
ZBASIC
dBASE II, III, III Plus
@RESULT
FRED
RESUME
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
dBASE III Plus
BASIC-APPLESOFT
ZBASIC
RETARGETABILITY
Ada
return
C
RETRY
dBASE III Plus
RETURN
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
CBASIC
BASIC-TI (extins)
COBOL
dBASE II, III, III Plus
Fortran
Modula-2
PL/I
RETURN NEXT
BASIC-TI (extins)
REUSABILITY
Ada
REVEAL
PC Forth
reverse
C
REVERSE
LISP
PL/I
REVISION
Ada
REVISION SET
Ada
REWIND
Fortran
Pilot
RFLACP
LISP
RGT
ABASIC
RIGHT
dBASE III Plus
BASIC-PLUS
ABASIC
Logo
RIGHT$
BASIC-PRAE
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
GW-BASIC
BASIC-APPLESOFT
CBASIC
ZBASIC
RMOVE
BASIC-aMIC

BASIC-80
RMDIR
GW-BASIC
RND
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
ZBASIC
COMAL
ROLL
Forth
ROT
BASIC-APPLESOFT
CBASIC
ZBASIC
Forth
ROTATE
BASIC-aMIC
BASIC-80
ROUND
dBASE III, III Plus
BASIC-AMSTRAD
PL/I
@ROUND
FRED
ROUNDED
COBOL
ROW
dBASE III, III Plus
RPLACA
LISP
RPLACD
LISP
RPO
PC Forth
RPT
BASIC-TI (extins)
RSET
BASIC-PLUS
BASIC-80
GW-BASIC
ZBASIC
RTRIM
dBASE III Plus
RUN
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
CBASIC
BASIC-APPLESOFT
ZBASIC
BASIC-TI (extins)
@RUN
FRED
RUN/!
dBASE III Plus

S

%s
C
/S
Super Calc
Visi Calc
SADD
CBASIC
Savo
Prolog
SAVE
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
BASIC-TI (extins)
BASIC-ATARI
CBASIC
ZBASIC
Logo
Forth
PC Forth
SAVEMEM
CBASIC
SAVEPICT
Logo
SAVE
dBASE II, III, III Plus
SCALAR TYPE
Ada
SCALE
BASIC-aMIC
BASIC-APPLESOFT
scanf
C
SCR
BASIC-aMIC
SCREEN
GW-BASIC
SCREEN$
BASIC HC-85, TIM S, SPECTRUM
SE
Logo
SEEK
dBASE III, III Plus
SEARCH
COBOL
PL/I

SEARCH ALL
COBOL
SECURITY
COBOL
SEG
ABASIC
SELECT
COBOL
@SELECT
FRED
SELECTED COMPONENT
Ada
SELECTION
Ada
SELECT
dBASE II, III, III Plus
SERIAL-IN
PC Forth
SERIAL NUMBER
Ada
SERIAL-OUT
PC Forth
SERIAL-STATUS
PC Forth
SET
COBOL
dBASE II, III
dBASE III Plus
LISP
Modula-2
RPG
@SET
FRED
SET ALTERNATE
dBASE II, III
dBASE III Plus
SET BELL
dBASE II, III
dBASE III Plus
SET CARRY
dBASE II, III
dBASE III Plus
SET COLON
dBASE II
SET COLOR
BASIC-ATARI
SET COLOR TO
dBASE II, III
dBASE III Plus
SET CONFIRM
dBASE II, III
dBASE III Plus
SET CONSOLE
dBASE II, III
dBASE III Plus
SET DATE
dBASE II
dBASE III Plus
SET DEBUG
dBASE II, III
dBASE III Plus
SET DEFAULT
dBASE II, III
dBASE III Plus
SET DELETED
dBASE II, III
dBASE III Plus
@SET DIRECTORY
FRED
SET DRIVE
FRED
SET ECHO
dBASE II, III
dBASE III Plus
SET EJECT OFF
dBASE II
SET EJECT ON
dBASE II
SET ESCAPE
dBASE II, III
dBASE III Plus
SET EXACT
dBASE II, III
dBASE III Plus
SET F
LISP
SET FORMAT
dBASE II, III
dBASE III Plus
@SET FORMULA
FRED
SET HEADING
Logo
dBASE II, III
dBASE III Plus
SET CATALOG
dBASE III Plus
SET CATALOG TO
dBASE III Plus
SET CENTURY
dBASE III Plus
SET DECIMALS
dBASE III, III Plus
SET DELIMITER
dBASE III, III Plus
SET DEVICE
dBASE III, III Plus
SET DOHISTORY
dBASE III Plus
SET FIELDS
dBASE III Plus
SET FIELDS TO
dBASE III Plus
SET FILTER
dBASE III, III Plus
SET FIXED
dBASE III, III Plus
SET FUNCTION
dBASE III, III Plus
SET HELP
dBASE III, III Plus
SET HISTORY
dBASE III Plus
SET HISTORY TO
dBASE III Plus

BASIC-APPLESOFT
CBASIC
ZBASIC
COMAL
Modula-2
PL/I
Super Calc

@SIN
FRED

SIND
PL/I

SINH
PL/I

@SIN (n)
Visi Calc

SYSTEM
GW-BASIC

SIZE
BASIC-TI (extins)
S-BASIC

SIZE-ERROR
COBOL

@SK
FRED

SKIP
dBASE II, III, III Plus

SLICE
Ada

SNAPSHOT CONTAINER
Ada

SOME
LISP

SORT
Multi Plan
dBASE II, III, III Plus

SOURCE-COMPUTER
COBOL

SOUND
BASIC-AMSTRAD
GW-BASIC

@SP
FRED

SPACE
dBASE III, III Plus
Forth
PC Forth

SPACE$
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC

SPACES
COBOL
PC Forth

SPACES$
ZBASIC

span
PC Forth

SPAN
Forth
PC Forth

SPC
BASIC-AMSTRAD
BASIC-PRAE
BASIC-COMMODORE
BASIC-80
GW-BASIC
ABASIC
ZBASIC

SPECIAL
RPG

SPEECH STRING TOO LONG
BASIC-TI (extins)

SPEED
BASIC-AMSTRAD
BASIC-APPLESOFT

SPLITSCREEN
Logo

SPO
Forth

SQ
BASIC-AMSTRAD

SQR
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
CBASIC
ZBASIC
COMAL

SQRT
Logo
Modula-2
PL/I
RPG
dBASE III, III Plus
Super Calc

@SQRT
FRED

@SQRT (n)
Visi Calc

squeeze
C

sscanf
C

SRG
ABASIC

@ST
FRED

$STACK
S-BASIC

STANDARD
COBOL
PC Forth

STATE
Forth
PC Forth

STATEMENT
Ada

static
- C

STATIC
- PL/I

STATUS
- BASIC-ATARI

@STD
- FRED

STEP
- BASIC-aMIC
- BASIC-PRAE
- BASIC HC-85, TIM S, SPECTRUM
- BASIC-AMSTRAD
- BASIC-COMMODORE
- BASIC-80
- BASIC-PLUS
- ABASIC
- GW-BASIC
- S-BASIC
- COMAL

STOCK
- BASIC-ATARI

STOP
- BASIC-aMIC
- BASIC-PRAE
- BASIC HC-85, TIM S, SPECTRUM
- BASIC-AMSTRAD
- BASIC-COMMODORE
- BASIC-80
- BASIC-PLUS
- ABASIC
- GW-BASIC
- BASIC-APPLESOFT
- CBASIC
- ZBASIC
- COBOL
- Fortran
- Logo
- PL/I

STORE
- BASIC-APPLESOFT
- LISP

STORE . . . TO . . .
- dBASE II, III, III Plus

STR$
- BASIC-aMIC
- BASIC-PRAE
- BASIC HC-85, TIM S, SPECTRUM
- BASIC-AMSTRAD
- BASIC-COMMODORE
- BASIC-80
- ABASIC
- BASIC-APPLESOFT
- CBASIC
- ZBASIC

STR
- dBASE II, III, III Plus
- GW-BASIC

stromp
- C

stropy
- C

streat
- C

strien
- C

STRIG
- BASIC-ATARI

STRING
- Ada
- COBOL
- Smalltalk

STRINGP
- LISP

STRING$
- BASIC-AMSTRAD
- GW-BASIC
- BASIC-PLUS
- ZBASIC
- BASIC-80
- ABASIC

strsave
- C

STRUCT
- Algol

STUFF
- dBASE III Plus

SUB
- BASIC-TI (extinš)
- RPG

SUBSTR
- dBASE III, III Plus

SUBCOMPONENT
- Ada

SUBEND
- BASIC-TI (extins)

SUBEXIT
- BASIC-TI (extins)

SUBPROGRAM
- Ada

SUBROUTINE
- Fortran

SUBR * *
- RPG

SUBSET
- LISP

SUBSTR
- PL/I

SUBSTRING
- LISP

SUBT
- LISP

SUBTRACT
- COBOL

SUBTYPE
- Ada

SUBUNIT
- Ada

SUCC
- Pascal

sum
- Prolog

SUM
- Super Calc
- dBASE II, III, III Plus

SUSPEND
- dBASE III Plus

@SUM
FRED
Visi Calc
@SUMDATE
FRED
SWAP
BASIC-AMSTRAD
BASIC-80
ABASIC
GW-BASIC
ZBASIC
Forth
switch
C
SWITCH
BASIC-PRAE
SYNCHRONIZED
COBOL
SYS
PC Forth
SYMBOL
BASIC-AMSTRAD

T

/T
Super Calc
Visi Calc
TAB
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
COMAL
TAG
BASIC-AMSTRAD
RPG
TAGOFF
BASIC-AMSTRAD
TAN
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
GW-BASIC
BASIC-APPLESOFT
CBASIC, ABASIC
ZBASIC
COMAL
PL/I
Super Calc
@TAN
FRED
Visi Calc
TAND
PL/I
TANH
PL/I
TARGET
Ada
TASK
Ada
TERPRI
LISP
TEST
BASIC-AMSTRAD
Logo
TESTB
RPG
TESTR
BASIC-AMSTRAD
TEXT
dBASE II, III, III Plus
S-BASIC
Text division
SAM 76
Text Functions
SAM 76
TEXTSCREEN
Logo
TEXTZ
RPG
TG
ABASIC
THEN
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
Algol
Forth
Modula-2
THING
Logo
@THOUSANDS
FRED
THROW
LISP
TI
BASIC-COMMODORE
TI$
BASIC-COMMODORE
TIB
Forth
PC Forth
TIME
dBASE III, III Plus
BASIC-AMSTRAD
ABASIC
PL/I
RPG

TIMES$
BASIC-AMSTRAD
GW-BASIC
@TIME
FRED
@TIME 1
FRED
@TIME 2
FRED
@TIME 3
FRED
TIMER
GW-BASIC
TIMES
ZBASIC
Prolog
@TM
FRED
TO
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
Logo
@TODAY
FRED
TOPLEVEL
Logo
TOTAL
dBASE II, III, III Plus
TRACE
BASIC-PRAE
Logo
@TRACE
FRED
$TRACE
S-BASIC
TRANSFER
Multi Plan
TRANSFORM
dBASE III Plus
TRANSPORTABILITY
Ada
TRANSLATE
BASIC-PLUS
PL/I
TRAP
BASIC-ATARI
TRIAD
PC Forth
TRIM
dBASE II, III, III Plus
TRN
BASIC-aMIC
TRM
ABASIC
TROFF
BASIC-AMSTRAD
BASIC-80
GW-BASIC
ZBASIC
TRON
BASIC-AMSTRAD
BASIC-80
GW-BASIC
ZBASIC
true
PC Forth
TRUE
Algol
Modula-2
PC Forth
TRUE PARENT
Ada
TRUNC
Modula-2
PL/I
TURTLESTATE
Logo
TYPE
dBASE II, III, III Plus
Ada
Forth
Modula-2
Pascal
PC Forth
Pilot
typedef
C
TYPE HANG
Pilot
TYPE PRINTER
Pilot

U

/U
Super Calc
U.
Forth
U>
Forth
UCASE$
CBASIC
UPDATE
RPG
UDAY
RPG
UM
Forth
UM/MOD
Forth
UMOUTH
RPG
@UNHIDE
FRED
@UNIT
FRED
UNLOCK
PL/I

UNPLOT
BASIC-aMIC
UNSAVE
BASIC-PLUS
UNLESS
BASIC-PLUS
UNT
BASIC-AMSTRAD
UNSPEC
PL/I
UNTIL
BASIC-PLUS
ABASIC
COBOL
COMAL
Forth
Modula-2
PC Forth
UPDATE
dBASE II, III, III Plus
Forth
PC Forth
UPPER$
BASIC-AMSTRAD
UPPER
dBASE III, III Plus
USABLE CONTAINER
Ada
USAGE
COBOL
USE
dBASE II, III, III Plus
Pilot
USE CLAUSE
Ada
USER
PC Forth
USING
BASIC-PRAE
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
USR
BASIC HC-85, TIM S, SPECTRUM
BASIC-COMMODORE
BASIC-80
GW-BASIC
BASIC-APPLESOFT
ZBASIC
UYEAR
RPG

V

/V
Visi Calc
VAL
BASIC-aMIC
BASIC-PRAE
BASIC HC-85, TIM S, SPECTRUM
BASIC-AMSTRAD
BASIC-COMMODORE
BASIC-80
dBASE II, III, III Plus
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-APPLESOFT
CBASIC
ZBASIC
Modula-2
VALUE
COBOL
Multi Plan
VALUE ENTRY
Visi Calc
VAR
Modula-2
S-BASIC
@VAR
FRED
VARIABLE
Ada
Forth
PC Forth
PL/I
2 VARIABLE
PC Forth
VARIANT
Ada
VARIATION SET
Ada
VARPTR
GW-BASIC
CBASIC
ZBASIC
VARPTR$
GW-BASIC
VARPTR # n
ZBASIC
VARYING
COBOL
PL/I
VERIFY
BASIC HC-85, TIM S, SPECTRUM
BASIC-COMMODORE
PL/I
VERSION
dBASE III Plus
VIEW
GW-BASIC
PC Forth
VIEWPORT
BASIC-aMIC
BASIC-80
VISIBILITY
Ada
VISIBLE PORT
Ada
@VLOOKUP
FRED
VOCABULARY
Forth
PC Forth

XOR
BASIC-AMSTRAD
BASIC-80
BASIC-PLUS
ABASIC
GW-BASIC
BASIC-TI (extins)
S-BASIC
Forth
PC Forth

XPOS
BASIC-AMSTRAD

Y

YCOR
Logo

YEAR
dBASE III, III Plus

YPOS
BASIC-AMSTRAD

@YEN
FRED

Z

/Z
Super Calc

Z-ADD
RPG

ZAP
dBASE III Plus

ZERO
BASIC-aMIC
COBOL

ZONE
BASIC-AMSTRAD
COMAL

Z-SUB
RPG

BIBLIOGRAFIE

1. Baltac, V. ş.a. – **Sisteme interactive şi limbaje conversaţionale,** Editura Tehnică, Bucureşti 1984.
2. Biondi, J., Clavel G. – **Introduction à la programmation,** Masson, 1981.
3. Connors, B., Edwards, S. – **Les grands classiques du jeu pour votre ZX-SPECTRUM,** Addison-Wesley Europe, 1984.
4. Cernian, O., Zamfirescu, P., Stan, St., Ionescu, A., Muşatescu, C. – **Limbajul BASIC VS, Programare aplicată pe noile calculatoare WANG,** Coeditare Editura Tehnică – Firma americană de calculatoare Wang, Bucureşti, 1986 (volum special din seria AMC).
5. Calter, P. – **Problem Solving with Computers, McGraw-Hill Book Company,** 1973.
6. Coan, J., – **Advanced BASIC, Applications and Problems,** Hayden Book Company, Inc., 1977.
7. Dumitraşcu, L., Pătruţ, St., Stan, St. – **Învăţăm FORTRAN... conversînd cu calculatorul,** Editura Tehnică, 2 vol., Bucureşti, 1981.
8. Dumitraşcu, L. – **Învăţăm COBOL... conversînd cu calculatorul,** Editura Tehnică, 2 vol., Bucureşti, 1985.
9. Dumitraşcu, L., Ioachim, Al. – **Tehnici de construire a programelor cu structuri alternative,** Editura Academiei R.S.R., 1981.
10. Dumitraşcu, L., Ioachim, Al. – **Generating FORTRAN programs for Decision Tables,** V.N.U., Utrecht, Olanda, 1988.
11. Dumitraşcu, L. – **BASIC pentru începători cu calculatorul personal,** A.M.C. nr. 48, Editura Tehnică, Bucureşti, 1986.
12. Dumitraşcu, L. – **103+15 calculatoare personale, Ghid de alegere şi utilizare (I),** A.M.C. nr. 52, Editura Tehnică, Bucureşti, 1986.
13. Dumitraşcu, L. – **103+15 calculatoare personale, Ghid de alegere şi utilizare (II),** A.M.C. nr. 53, Editura Tehnică, 1986.
14. Dumitraşcu, L., Manea, M., Marinoiu, C. – **Programare. Note de curs,** I.P.G., (Uz intern), Ploieşti, 1987.
15. Damian Iordache, N. – **Metode şi tehnici de programare structurată,** vol. I, Academia „Ştefan Gheorghiu", CPADC (Uz intern), Bucureşti, 1979.
16. Dumitrescu, St. – **Programare şi metode numerice,** I.P.G. (Uz intern), Ploieşti, 1987.
17. Dwyer, Th., Critchfield, M. – **BASIC and the Personal Computer,** Addison-Wesley Publishing Company, 1986.
18. Erskine, R., Walwyn, H., Stanley, P., Bews, M. – **Sixty Programs for the AMSTRAD CPC464,** London, 1984.
19. Franz, E. – **The Great Book of Games; 46 Programs for the Commodore-64,** Elcom Publishing, 1986.
20. Guţu, St., Dumitraşcu, L. – **Elemente de inteligenţă artificială pentru conducerea operativă a producţiei,** Editura Academiei R.S.R., 1981.
21. Garry, M. – **Programming with Graphics,** Granada Publishing Limited Frogmore, St. Albans, 1986.
22. ΧΡΗΣΤΟΥ ΚΟΙΛΙΑ, Η ΓΛΩΣΣΑ **BASIC** ΚΑΙ ΟΙ ΕΦΑΡΜΟΓΕΣ ΤΗΣ, ΑΘΗΝΑ, 1986
23. Gottfried, B. – **Programming with BASIC, Including Microcomputer BASIC,** 2/ed, Schaum's Outline Series in Computers, McGraw-Hill Book Company, 1986.
24. Hartnell, T., Jones D. – **La conduite du ZX SPECTRUM,** Eyrolles, 1983.
25. Harwood – **Jeux et applications pour ZX SPECTRUM,** Eyrolles, 1983.
26. Lazea, H., Davidescu, C., Lorenţ, Th., Făgărăşeanu, I. – **Fond de instruire asistată de calculator pentru BASIC-PLUS,** Manual de referinţă, Academia „Ştefan Gheorghiu", Centrul de prelucrare automată a datelor şi consultanţă, Întreprinderea de calculatoare electronice, Bucureşti, 1982.
27. Lien, D. – **The BASIC Handbook,** 2nd Edition, Encyclopedia of the BASIC Computer Language, COMPUSOFT, San Diego, 1984.
28. Larsen, S. – **Sprite Graphics, Commodore 64,** Prentice-Hall, 1986.

29. Maynard, J. – **Computer Programming, Made Simple,** Heinemann, London, 1983.
30. Miller, A. – **BASIC Programs for Scientists and Engineers,** SYBEX, 1981.
31. Marinoiu, V., Dumitrașcu, L., Minoiu, Şt., Macri, I., Popa, C., Marinoiu, C., Pătrășcioiu, C. – **Programare, Îndrumar de laborator,** Ploiești, 1987.
32. Pignelet, P. – **BASIC et extension,** Masson et EAP, Paris, 1980.
33. Petrescu, A. ş.a. – **Totul despre... calculatorul personal aMIC,** Editura Tehnică, 2 vol., Bucureşti, 1985.
34. Petrescu, A. ş.a. – **Microcalculatoarele FELIX M18, M18B, M118,** Editura Tehnică, Bucureşti, 1984.
35. Popa, C. – **Programare şi metode numerice,** I.P.G. (Uz intern), Ploieşti, 1978.
36. Patrubany, N., Pop, B., Kiss, A., Socaciu, N., Szász, D. – **Familia de calculatoare personale româneşti PRAE şi limbajul său BASIC,** A.M.C. nr. 51, Editura Tehnică, Bucureşti, 1985.
37. Rupp, W., Hartman, P. – **Commodore 64, Game Construction Kit,** DATAMOST, 1987.
38. Simpson, H. – **Serious Programming in BASIC,** TAB BOOKS Inc., 1986.
39. Vuldy, I. – **Graphisme 3D sur votre microordinateur,** EYROLLES, Paris, 1985.
40. Wintermeyer, L. – **Applesoft BASIC Toolbox,** Addison-Wesley Publishing Campany, 1984.
41. Wynford, J. – **BASIC Programming on the Amstrad,** Micro Press, 1985.
42. * * * – **64 Sound and Graphics, Computer's First Book of Commodore 64,** Greensboro, North Carolina, 1986.
43. * * * – **Compute '!'s Gazette, for COMMODORE Personal Computer Users,** 1987.
44. * * * – **Radio Shack, BASIC Computer Language, It's easier than you think!,** 1977.
45. * * * **TIM S, Microcalculator personal, Manual de funcţionare şi utilizare,** I.T.C.I., Timișoara, FMECTC, 1987.
46. * * * – **MSGW-BASIC Interpreter under MS-DOS,** User Guide, Olivetti Personal Computer, 1984.
47. * * * – **Science & Vie Micro,** nr. 31, 32, 33, 34, 35, 36/1986, 1987.
48. * * * – **CP/M, BASIC-80,** Întreprinderea de Echipamente Periferice Bucureşti, 1986.
49. * * * – **ZX SPECTRUM SINCLAIR,** Introduction, DRION COMPUTERS, 1986.
50. * * * – **BASIC AMSTRAD, AMSOFT,** 1985.
51. * * * – **BASIC-PLUS, Manual de operare,** Institutul de Cercetări pentru Tehnică de Calcul, 1983.
52. * * * – **ABASIC, Interpretor BASIC sub ARIEL, Manual de prezentare, operare şi utilizare,** 1984, decembrie, București, Institutul Central pentru Conducere şi Informatică, Centrul de Calcul.
53. * * * – **BASIC-PLUS-2, RSX-11M/IAS, User's Guide,** 1978.
54. * * * – **Dicţionarul explicativ al limbii române,** Editura Academiei R.S.R., Bucureşti, 1975.
55. * * * – **Dicţionar de informatică,** Editura Ştiinţifică şi Enciclopedică, Bucureşti, 1981.
56. Ioachim, Al. – **Cercetarea şi proiectarea asistată de calculator. Elemente teoretice şi practice,** vol. 1, 2, 3; 1986, 1989, Academia de partid pentru învăţămînt social-politic (uz intern), București.
57. Encarnaçao, J., Schlechtendal, D. – **Computer Aided Design, Fundamentals and System Architectures,** Springer – Verlag, Berlin, Heidelberg, New-York, Tokyo, 1983.
58. Rogers, D., Adams, I. – **Mathematical Elements for Computer Graphics,** Mc Graw – Hill Book Company, New-York.
59. Dargery, Y. – **CP/M Plus sur Amstrad 6128 et 8256,** Editions du P.S.I., 1986.
60. * * * – **Sistemul de operare DOS-PC,** Institutul de cercetare ştiinţifică şi inginerie tehnologică pentru tehnică de calcul şi informatică, sector Tehnică de calcul, Cluj-Napoca, 1988.
61. Birnes, W., Hayfield, N. – **Personal Computer Programming Encyclopedia, Languages and Operating Systems,** Mc Graw Hill, 1985.
62. * * * – Seria continuă **A.M.C.,** Editura Tehnică, Bucureşti, (1984–1989, sub tipar).
63. Bihan, P. – **Programmer en QUICK-BASIC,** Eyrolles, 1988.
64. Bihan, P. – **Programmer en TURBOBASIC,** Eyrolles, 1987.
65. Vanryb, B., Politis, R. – **BASICA et GW-BASIC MICROSOFT,** Eyrolles, 1988.
66. Krutch, J. – **Experiences d'intelligence artificielle en BASIC,** Eyrolles, 1985.
67. Monteil, M., Schomberg, R. – **Programmes d'intelligence artificielle en BASIC,** Eyrolles, 1985.
68. Dumitraşcu, L., Marinoiu, Cr. – **Programarea în dBASE II. Proiectarea şi realizarea unei microbaze de date – studiu de caz,** I.P.G., Ploieşti, 1989.

VOL. 1 + VOL. 2 : 97 LEI

TOTUL DESPRE...

BASIC

..IN 14 CONVERSATII SI 7 SINTEZE

☐ Se abordează mai toate tipurile de probleme adecvate familiei de limbaje: calcule numerice, alfanumerice, vectoriale, matriciale, lucrul cu toate tipurile de fișiere, grafică bi- și tridimensională, lucrul cu meniuri, ferestre, culori, muzică, simularea mișcărilor, animație, conversațiile incheindu-se cu un studiu de caz complex.

☐ Sintezele, ce ocupă volumul 2, sint construite (cu excepția sintezei 20, ce conține programele exemplelor din conversații) in maniera unor tezaure dense, de consolidare, completare și regăsire a cunoștințelor insușite din volumul 1 ca și de extindere a lor asupra componentelor, echipamentelor, software-ului de sistem, aplicațiilor, programării, in general, și programării in BASIC, in special.

☐ Microprocesoare, sisteme de operare, codificări in 20 limbaje de nivel inalt și in 10 produse program generalizabile, un dicționar comparativ al acestora, proiectare asistată, reprezentări geometrice, jocuri, memento-uri complexe pe echipamente electronice tipice, ș.a.

☐ O excelentă carte de autoinstruire și de aplicare practică pe o multitudine de echipamente (v. și grafica), adusă la zi și prin pașii dintre conversații și sinteze.

M 118 | FELIX C

CORAL | a MIC | JUNIOR

FELIX PC | SPECTRUM | AMSTRAD

COMMODORE | INDEPENDENT | TPD

ISBN 973-31-0010-2
ISBN 973-31-0012-9